The Secure MVC pattern

Nelly Delessy-Gassant
Grambling State University, Grambling, LA, USA, gassantn@gram.edu

Eduardo B. Fernandez
Florida Atlantic University, Boca Raton, FL, USA, ed@cse.fau.edu

ABSTRACT

The Model-View-Controller (MVC) pattern has been used in many frameworks and applications for the last few decades. Now, with advent of the pervasive web, the MVC pattern is applied in a large variety of distributed applications. However, the use of the web also exacerbated the security threats faced by such applications. In this paper, we present the Secure MVC pattern, which describes how to add security to the MVC pattern. We show how several fundamental security patterns such as the Authenticator, Role-Based Access Control (RBAC), the Secure Channel, and Secure Logging must be applied to the MVC components. In addition it is necessary to add new components for sanitizing incoming data.

Keywords: security pattern, pattern, MVC

1. INTRODUCTION

The Model-View-Controller (MVC) pattern has been used in many frameworks and applications for the last few decades. Now, with advent of the pervasive web, the MVC pattern is applied in a large variety of distributed applications. However, the use of the web also exacerbated the security threats faced by such applications. There is a great deal of design pattern advice on how to build distributed systems, e.g. (Buschmann et al., 1996, Kircher and Jain, 2004, Schmidt et al.). There is also a great deal of experience with securing distributed systems, e.g., (Kaufman et al., 2002). However, much of the experience gained in securing distributed systems has not worked its way back into the design patterns. In this paper, we present the Secure MVC pattern, which describes how to add security to the MVC pattern.

2. SECURE MVC

2.1 INTENT

Add security to the interactions of users with systems as defined in MVC

2.2 EXAMPLE

eLeague is a company that provides tournament management applications to a variety of sport leagues. The company develops service-based mobile applications allowing sport administrators, athletes and coaches to view and/or maintain their teams, schedules and scores from anywhere. The user interface of the application is susceptible to change often, as it has to adapt to new generations of mobile devices, whereas the structure of a tournament's information changes less often. In addition, tournament information is sensitive and should be modified only by authorized users.

2.3 CONTEXT

The Model-View-Controller (MVC) pattern (Buschmann et al., 1996) provides a way to add modularity to an application by separating its functionalities into three loosely-coupled components, the Model, the View and the Controller, thus rendering the entire application more maintainable. This pattern has long been applied to both standalone applications and distributed systems. Now, the MVC pattern is widely used in web applications ranging from service-based applications to mobile web applications. Figure 1 shows the structure of the MVC pattern.

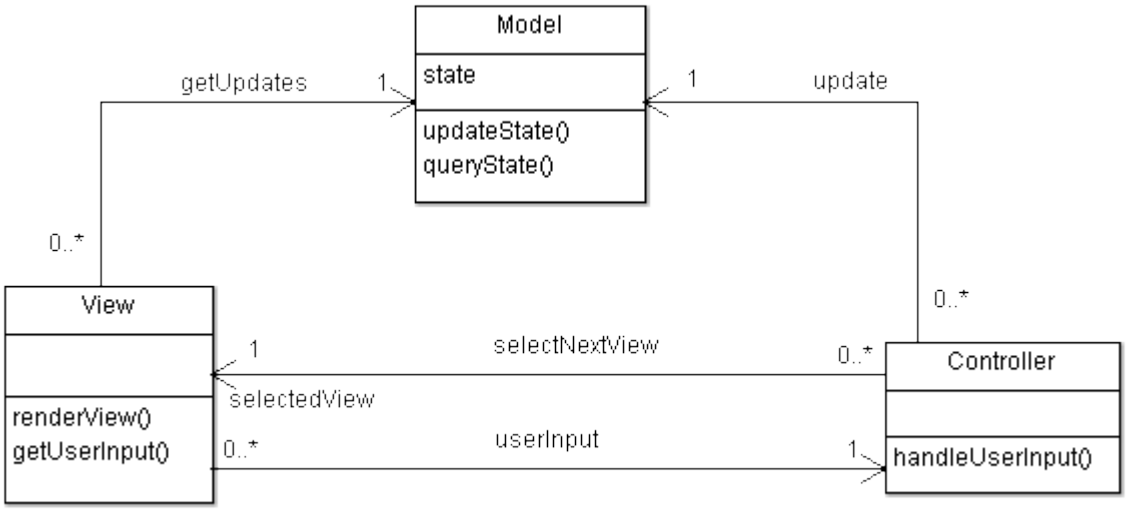

Figure 1: Structure of the MVC pattern

Systems applying the MVC pattern are typically multi-user systems and their Model should be accessible and /or modifiable only by certain categories of users. At the same time, the use of the web as a transport layer has brought some new threats that must be mitigated: eavesdropping, impersonation via session hijacking, unauthorized modification via such attacks as SQL injection, or cross site scripting.

2.4 PROBLEM

How to keep an acceptable level of security among the Model, the View and the Controller in the presence of possible attacks?

This problem is affected by the following forces:

- **Authenticity**. We need to be sure that users who interact with our system are legitimate. Remote users will want to be sure that our system is authentic.
- **Confidentiality**. We may need to restrict access to the Model's information only to some users or roles. Also some portions of the data in transit from the Model to the View must be protected against eavesdropping.
- **Integrity**.We may want to allow only some users or roles to make changes to the Model, and only authorized changes.
- **Records**. The model may contain sensitive information and we want to have a record of all accesses to it.

2.5 SOLUTION

The Secure MVC pattern allows users to securely access and/or modify sensitive information located in the Model component. Basic security patterns are applied to provide authentication, authorization, secure communications and logging. In addition, it might be necessary to sanitize incoming and/or outgoing data in order to prevent malicious payload attacks such as SQL injections, or cross site scripting attacks.

Access control can be added at the Model level and/or at the Controller level. Adding access control at the Controller level is less intrusive for the Model, however, it is coarse-grained. Access control at the Model level can be finer-grained and could be based on specific attributes of the Model.

2.5.1 STRUCTURE

Security is added to the MVC pattern by applying several security patterns (c.f. Figure 2). An **Authenticator** (Schumacher, 2006) identifies and authenticates the **User** requesting access to the **Model** and/or to the **Controller** through the **View**. We apply the Role-based Access Control (RBAC) pattern (Schumacher, 2006) to the MVC pattern. **Users** are members of **Roles** and **Rights** are assigned to **Roles**. A **Right** defines the access type that can be applied by a role to the **Model**, to the **Controller**, or to both. A **Reference Monitor** (Fernandez and Pan, 2001) enforces the access control rights defined in the RBAC pattern.
A **Secure Channel** (Braga et al., 1998) is responsible for securing traffic between the **View** and the **Model** and between the **View** and the **Controller**. Since communications between the **Controller** and the **Model** do not typically use the web, it is not necessary to secure communications between those components.
A **Security Logger** records all security-sensitive actions on the model. Finally, incoming data and outgoing data must be filtered by a **Data Sanitizer** to prevent malicious payloads embedded into otherwise authorized requests from accessing the model.

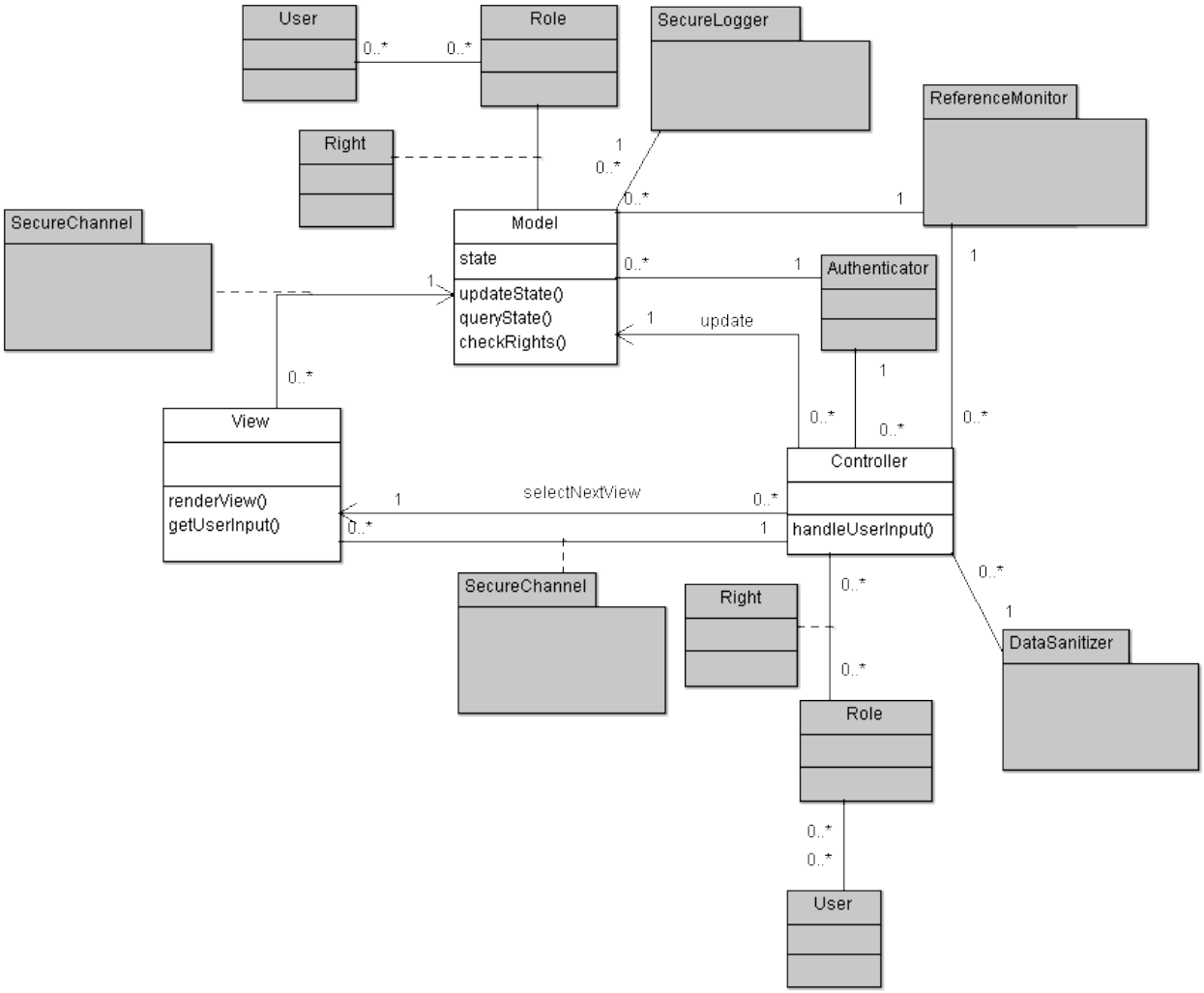

Figure 2: Structure of the Secure MVC pattern

2.5.2 DYNAMICS

Figure 3 shows a UML Sequence Diagram in which an authenticated enters some input through the currently displayed **View**. The data embedded in the request sent to the **Controller** is first inspected by the **DataSanitizer**, then the request is intercepted by the **ReferenceMonitor**, and if a **Right** exists for the requested operation on the **Controller**, this latter can handle the user input. The **Controller** handles a user event by updating the **Model** and selecting another **View**. The request for updating the **Model** can also be intercepted by the **ReferenceMonitor** for finer access control. The access control decision is recorded by the **SecurityLogger**. Finally the selected **View** gets data from the **Model**. The request is again intercepted by the **ReferenceMonitor**.

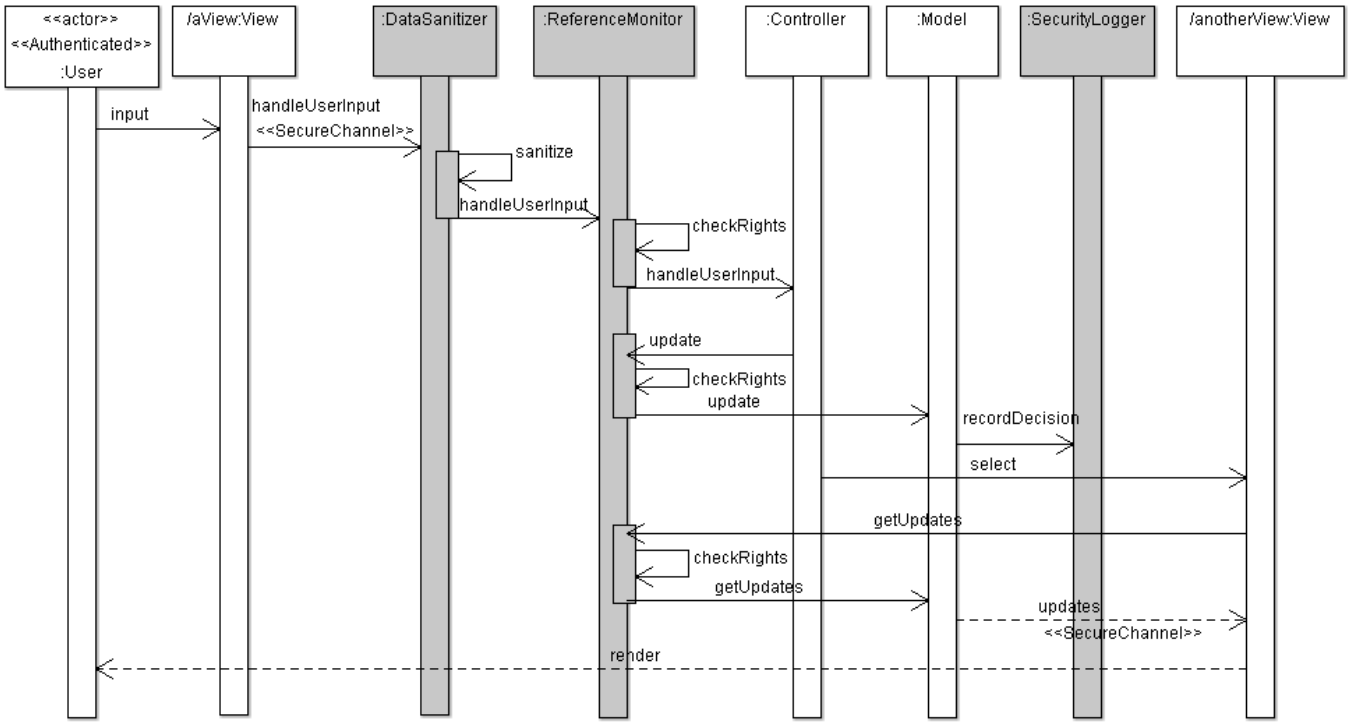

Figure 3: UML Sequence Diagram for the Secure MVC pattern

2.6 IMPLEMENTATION

In order for the **View** to obtain data updates from the **Model**, it is possible to use push or pull. In an asymmetric context such as the web, the **View** has no choice but to constantly poll the **Model**. In (Stratmann et al., 2011), a method called long poll is used. In standalone applications, the Observer pattern can be applied (Gamma et al., 1994).

2.7 KNOWN USES

- The ASP .Net MVC framework (Microsoft Corp., 2011) provides authentication and authorization functionalities at the controller's method level. In addition, AntiXSSLibrary and HtmlSanitizationLibrary are two libraries that can be used for protecting against XSS (Cross-Site Scripting) attacks as well as CSRF or XSRF (Cross-Site Request Forgery) attacks.
- The Struts web framework (The Apache Software Foundation, 2012) provides a Validation Framework, which is the primary method of validating a struts based application. Output sanitation is the process of ensuring that your output does not contain HTML or XML specific characters. In addition, roles can be mapped to ActionMapping objects, a Controller component, and authentication attributes can be specified. The framework supports multiple authentication schemes, such as password authentication, FORM-based authentication, authentication using encrypted passwords, and authentication using client-side digital certificates. SSL can be enabled as a secure transport layer.

- Spring Web MVC framework (Springsource Community, 2012) offers similar security features.

2.8 CONSEQUENCES

- **Authenticity**. An authentication system may confirm to the users that they are talking to the right model. Conversely, authentication indicates that the users accessing the model are legitimate.
- **Confidentiality and Integrity.** An authorization system can enforce confidentiality and integrity. A secure communication protocol ensures that data in transit remains confidential. The integrity of the system is also guaranteed by the data sanitizer which eliminates attacks from malicious embedded payloads.
- **Records.** A security Logger (Fernandez et al., 2011a) can record all security-sensitive actions.

2.9 EXAMPLE RESOLVED

eLeague can apply the Secure MVC pattern. The following roles are defined: administrator, coach, athlete, and each individual user are asked to register to the system so that they can be authenticated whenever they need to access the system. Each operation of the controller can be mapped to a set of authorized roles and for accessing sensitive data, it can be required to use a secure transport protocol. Those rules are enforced by a trusted component of the system.

2.10 RELATED PATTERNS

- Authorization is enforced by a Reference Monitor (Fernandez and Pan, 2001).
- The rights structure can follow an RBAC pattern (Schumacher, 2006).
- Authentication is performed by means of the Authenticator pattern (Schumacher, 2006).
- Logging can be done using a Security Logger (Fernandez et al., 2011a).
- The Secure Channel pattern is used to secure the communications channels (Braga et al., 1998). It supports the encryption/decryption of data. Another version is given in (Schumacher, 2006).
- The relationship between Views and the Model can use the Secure Publish Subscribe model (Fernandez et al., 2011b). Views could subscribe to updates from the Model.
- The MVC pattern can be structure according to the Secure 3-Tier Architecture (Fernandez et al., 2008).

3. CONCLUSION

In this paper, we presented the Secure MVC pattern, which describes how to add security to the MVC pattern. We show how several fundamental security patterns such as the Authenticator, Role-Based Access Control (RBAC), the Secure Channel, and Secure Logging must be applied to the MVC components. The Secure MVC pattern allows users to securely access and/or modify sensitive information located in the Model component. Basic security patterns are applied to provide authentication, authorization, secure communications and logging. In addition, it might be necessary to sanitize incoming and/or outgoing data in order to prevent malicious payload attacks such as SQL injections, or cross site scripting attacks. Access control can be added at the Model level and/or at the Controller level. Adding access control at the Controller level is less intrusive for the Model, however, it is coarse-grained. Access control at the Model level can be finer-grained and could be based on specific attributes of the Model.

4. ACKNOWLEDGEMENTS

REFERENCES

Braga, A., Rubira, C., and Dahab, R. (1998). "Tropyc: A pattern language for cryptographic object-oriented software". Chapter 16 in *Pattern Languages of Program Design* 4 (N. Harrison, B. Foote, and H. Rohnert, Eds.). Also in *Procs. of PLoP'98*, DOI= http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/

Burbeck, Steve. (1992). "Application Programming in Smalltalk-80: How to use Model-View-Controller (MVC)."*University of Illinois in Urbana-Champaign (UIUC) Smalltalk Archive.*: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html, 03/06/12. (date accessed)

Buschmann, F.,Meunier, R., Rohnert, H., Sommerlad, P., Stal., M. (1996*). Pattern-Oriented Software Architecture: A System of Patterns*, Volume 1, Wiley, 1996.

Fernandez, E. B. and Pan, R.. (2001). " A Pattern Language for security models", *Procs. of the 8th Annual Conference on Pattern Languages of Programs (PLoP 2001)*, 11-15 September 2001, Allerton Park Monticello, Illinois, USA, 2001. http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions

Fernandez, E. B. and Larrondo Petrie, M. M. (2007). "Securing design patterns for distributed systems", Chapter 3 in *Security in Distributed, Grid, and Pervasive Computing*, Y. Xiao (Ed.)., Auerbach Pubs., Taylor & Francis Group, LLC, 2007, 53-66.

Fernandez, E. B., Fonoage, M., VanHilst, M., Marta, M.. (2008). "The Secure Three-Tier Architecture Pattern," cisis, pp.555-560, *Procs. of the International Conference on Complex, Intelligent and Software Intensive Systems*, 2008

Fernandez , E. B., Mujica, S., and Valenzuela, F. (2011). "Two security patterns: Least Privilege and Secure Logger/Auditor.", *Procs.of Asian PLoP 2011*.

Fernandez, E. B., Yoshioka, N., and Washizaki, H.. (2011) "Two patterns for distributed systems: Enterprise Service Bus (ESB) and Distributed Publish/Subscribe*", Procs. of the 18th Conference on Pattern Languages of Programs* (PLoP 2011)

Fowler, M.. (1997). *Analysis patterns -- Reusable object models*, Addison- Wesley, 1997.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design patterns: elements of reusable object-oriented software*, Boston, Mass:Addison-Wesley, 1994.

Kaufman, C., Perlman, R., and Speciner, M.. (2002). *Network Security 2nd ed.*, Prentice-Hall 2002

Kircher, M. and Jain, P., (2004) *Pattern-oriented software architecture, vol. 3 , Patterns for Resource Management*, J. Wiley, 2004.

Microsoft Corp. (2011). The ASP .Net MVC framework, http://www.asp.net/mvc/mvc3, 03/06/12. (date accessed)

Schmidt, D., Stal, M., Rohnert, H., and Buschmann, F., *Pattern-Oriented Software Architecture*, John Wiley & Sons, West Sussex, England, 2000.

Schumacher, M., Fernandez, E. B., Hybertson, D., Buschmann, F. and Sommerlad, P. (2006) *Security Patterns: Integrating security and systems engineering"*, Wiley 2006. Wiley Series on Software Design Patterns.

Springsource Community. (2012). The Spring Web MVC framework, http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html, 03/06/12. (date accessed)

Stratmann, E., Ousterhout, J. and Madan, S.. (2011). Integrating long polling with an MVC framework. In *Proceedings of the 2nd USENIX conference on Web application development*(WebApps'11). USENIX Association, Berkeley, CA, USA, 10-10.

The Apache Software Foundation. (2012). The Struts web framework, http://struts.apache.org/, 03/06/12. (date accessed)

Authorization and Disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.