

Two patterns for web services security

Eduardo B. Fernandez
Dept. of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431, USA

Abstract

Patterns are widely used in software engineering where they have been successful in improving analysis and design by encapsulating the experience of many designers. Security patterns are a recent development as a way to encapsulate the accumulated knowledge about secure systems design. We present here two patterns for web services: 1) a Security Assertion Coordination pattern that coordinates authentication and authorization using a Role-Based Control (RBAC) model for access to distributed resources; and 2) A pattern for XML firewalls, that filters XML messages or documents according to institution policies. Because of space restrictions we only describe some sections of the standard template descriptions; more details can be seen in [1] and [7].

Keywords: Distributed systems security, Object-oriented patterns, SAML, Security patterns, XML firewalls

1.0 Introduction

Web services are increasing in importance but one of the main obstacles to their acceptance is security. This situation is improving with the appearance of several security standards that define architectural requirements to provide appropriate levels of security [2]. One of these standards is the Security Assertion Markup Language (SAML), that expresses security assertions that can be exchanged between nodes in a distributed system. SAML defines an underlying security model based on the access matrix with a reference monitor [sam]. There is a Policy Decision Point (PDP), which corresponds to the authentication and authorization rules (or policies). Enforcement of access requests for resources is done by a Policy Enforcement Point (PEP), which applies the defined policies to the request. Because the abstract security architecture defined by SAML is a standard we can describe it conveniently by a pattern. Another important architecture for secure web services is the XML firewall, which controls access to web services by XML documents. This also can be described abstractly by a pattern. Patterns are widely used in software engineering where they have been successful in improving analysis and design by encapsulating the experience of many designers. In analogy, security patterns are a recent development as a way to encapsulate the accumulated knowledge about secure systems design [4]. Similarly to software patterns, security patterns are described using a fixed template. We present here two patterns for web services: 1) a Security Assertion Coordination pattern that coordinates authentication and authorization using a Role-Based Control (RBAC) model for access to distributed resources; and 2) A pattern for XML firewalls, that filters XML messages or documents according to institution policies. Because of space restrictions we only describe some sections of the standard template descriptions; more details can be seen in [1] and [7].

2.0 The Role-Based Security Assertion Coordinator

2.1 Intent

The Role-based Security Assertion Coordinator pattern allows seamless exchange of security data among organizations in a distributed environment in order to coordinate role-based access control to protected resources.

2.2 Context

A distributed environment including heterogeneous systems and web services.

2.3 Problem

Current systems lack feasible solutions to the problem of providing precise access control to resources, often requiring custom-built approaches that may not be easy to upgrade or modify. The growth of the number of networked business partners and their processes requires a means to exchange security information in a standardized format that is flexible to change at the same time. Costs are involved in custom integration processes, where time becomes crucial in achieving a quicker time-to-market competitive advantage. Costs include developer cost and development time. The security of the shared data becomes another concern. Consistency of data exchange has to be assured. Interoperability of systems across various implementation platforms stands as a significant obstacle. Adding a new layer of security verification policies often proves tedious and costly in the current systems.

2.4 Forces

A solution to the problem is affected by the following forces:

- --Distributed systems are in great need of integrating their inner processes that share commonly used data. Exchange of security related data in particular poses an important problem when the issues of interoperability is of concern. Organizations must be able to easily add new security layers across the distributed environment with little changes.
- Distributed environments must not resort to expensive global custom code changes in order to reflect new changes in security policies or data structure.
- Organizations in the distributed environment must have the ability to quickly achieve higher, more refined levels of security data control for better adherence to the continuously changing nature of organizational business rules.
- Each online destination site often has its own custom-made authentication or authorization system. This means that decisions involved in selecting hardware, operating system, or software platform may be biased toward local organizational factors. However after the deployment of the site, the need to integrate the system with other business partners may require to redesign the authentication or authorization model.
- RBAC is the most commonly used model for web-oriented systems. It should be possible to use this model across node boundaries. Assigning roles to individual online users is another need, especially if such information needs to be exchanged with minimal effort [3].

2.5 Solution

Exchange security information using a standard. In particular, manage security data in the form of XML-based SAML assertions using the SOAP protocol over HTTP. SAML has the benefit of easily implementing Role-Based Access Control with the help of attribute assertions that can be verified by the communicating parties. This solution is also independent of what platforms are implemented across the organization. Moreover, by the nature of SAML as an XML standard, extensibility of security data structures and its exchange between involved parties is convenient. Figure 1 shows the class diagram for this pattern. We have left out dynamic aspects (see [7]).

The *AssertionCoordinator* is the main class responsible for the creation and distribution of *Assertions* to requesting parties. This class uses the *UserAccount* to retrieve information needed to create the SAML

assertions. Since *Assertions* are in SAML XML format, they can be transferred easily over HTTP using SOAP. A web portal is an example of an *AssertionCoordinator* that maintains private information about customers who use services offered by partner sites, modeled in this pattern as the *PolicyDecisionEnforcers*. *Clients* who are authenticated against the web portal's databases become *Principals* eligible to access the services offered by the web portal's partners, the *PolicyDecisionEnforcers*. When *Principals* make service requests to *PolicyDecisionEnforcers*, the *AssertionCoordinator* on the web portal would issue *Assertions* on behalf of the authenticated *Principal*. The *Assertions* are exchanged with the *PolicyDecisionEnforcers* that, upon further local verification on their part, grants or revokes access to the requested resource. *Assertion* objects in this pattern are characterized by role attributes. This enables finer control over security verification policies on the side of the *PolicyDecisionEnforcers*. Each *Assertion* has an *AssertionAttribute* that stores the role information for the authenticated *Principal*. When the *PolicyDecisionEnforcer* receives the *Assertion* from the *AssertionCoordinator*, it extracts the *Principal's* role information represented as an *AttributeAssertion* within the requested *Assertion*. Role could be a qualitative identifier and could define a criterion that best suits the business need. So if *PolicyDecisionEnforcer* defines access to protected resources based on membership level defined as standard, premium, gold or any other role-based criterion, such information would be expected to be encoded in the SAML XML structure of the *AttributeAssertion* associated with the current *Principal's Assertion*.

2.6 Consequences

The Role-Based Security Assertion Coordinator patterns offer a number of *benefits*:

- *Centralized data exchange*. This centralized pattern for exchanging secure data across a distributed environment is based on assertions exchanged between entities in this framework and the *AttributeCoordinator*. Requests for protected resources and the responses issued by the protecting entity are made through these assertions.
- *Standardized approach*. The fact that data the communicate parties in this pattern share a standard communication protocol, namely, SAML assertions, for their security data exchange purposes means that the underlying platforms of the implemented systems are not affected by changes in policies or data structure.
- *Role-based access*. Entities in a distributed environment can now regulate access rights not just by username and password, but also have the added feature of filtering access base on roles.
- *Extensibility*. Although through this pattern we have shown how role-based access can be used as an attribute assertion in order to enforce a finer control over security policies, organizations using the same pattern could substitute the role attribute for other attributes that they may see more fitting.

The Role-Based Security Assertion Coordinator patterns has these *liabilities*:

- *Complex to implement*. Since SAML, and therefore the pattern, depends upon XML signatures and their verification when establishing trust relationships between distributed parties, a level of complexity is involved with the initial setup of the *AssertionCoordinator* pattern.
- *Computationally expensive*. Since each transaction is usually associated with the creation of an assertion, large volumes of transactions may become a system burden. Moreover, the XML parsing time spent verifying signatures while establishing trust also adds to the computational processing cost.

2.7 Known uses

Examples include: RSA ClearTrust; AssureAccess [2] and IONA's platform for Web Services [8].

2.9 Related patterns

This pattern follows a general approach using layers to coordinate security . There are patterns for RBAC [4] and for other aspects of security which are relevant here.

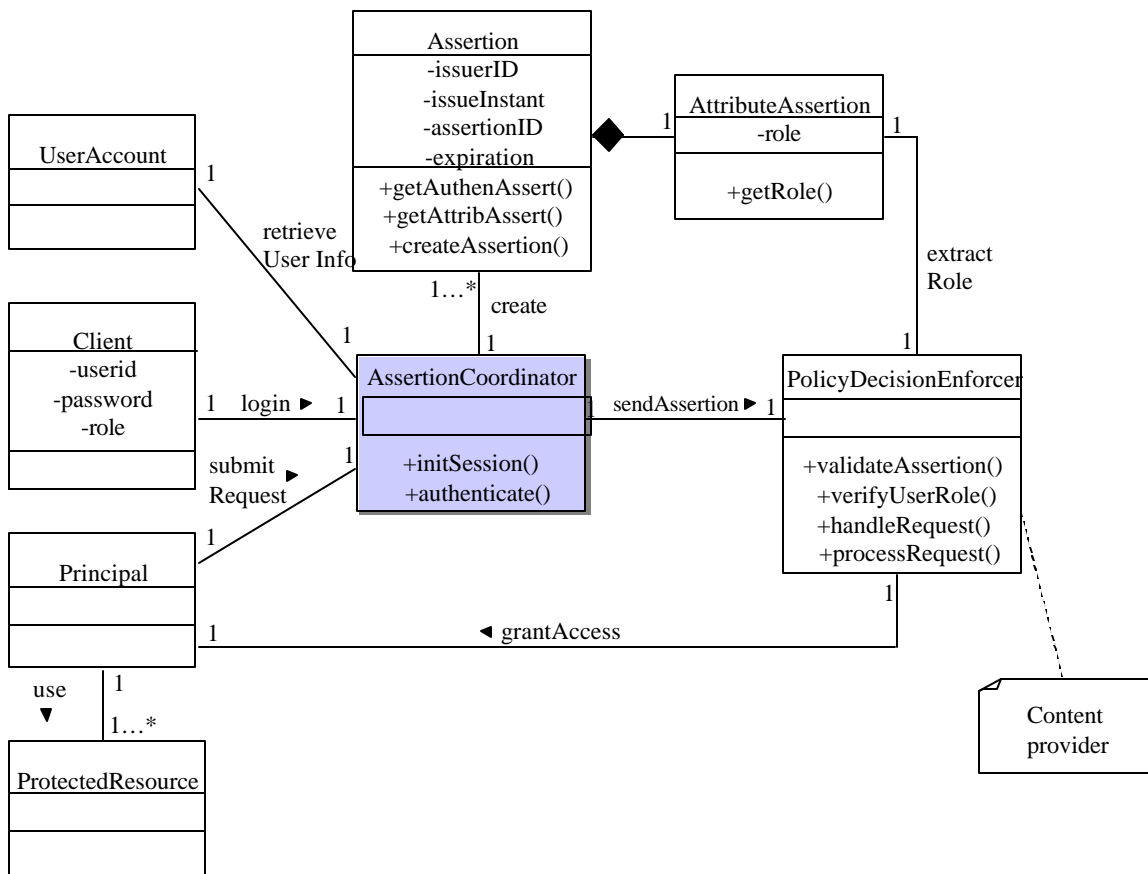


Figure 1. Class diagram of the Role -Based Assertion Coordinator

3.0 The XML Firewall pattern

3.1 Intent

To filter XML messages to/from user-defined applications, based on the business access control policies and the content of the message.

3.2 Context

User-defined applications executing in distributed systems accessed through a local network, from the Internet, or from external networks. These applications communicate through XML messages and could be web services or applications using web services. Specific security policies have been defined by the institution, expressed as authorization rules (policies).

3.3 Problem

Some user-defined applications use tunneling into authorized flows (HTTP, SMTP,...) to communicate with the outside. They use higher level protocols such as SOAP and communicate through XML documents. The XML documents in these messages can contain harmful data and can be used to perform attacks against applications. Moreover, network firewalls provide infrastructure security but become useless when these high level protocols and formats are used.

3.4 Forces

- Document formats are subject to change, some new ones may appear (XML dialects); the firewall must adapt easily to these changes.
- New types of harmful data may be used by attackers, the firewall must adapt easily to these new types of attacks.
- There are many ways to filter, we need to separate the filtering code from the application code.
- There may be numerous applications, that may require different levels of security.
- New applications may be integrated into the system after the firewall has been put into operation. This integration should not require additional costs.

3.5 Solution

A client can access a service of an application only if a specific policy authorizes it to do so and if the content of the message sent is considered to be safe for the applications. Policies for each application are centralized within the Application Firewall, in a *PolicyDefinitionPoint*. Each application is accessed by a client through a *PolicyEnforcementPoint*, that enforces the access control for the applications, by authenticating the client through its identity data stored in the *PolicyDefinitionPoint*, looking for a mapping policy for the request and checking the content of the message. Its structure is validated through a database of valid XML schemas, and the data it conveys is checked through a *HarmfulDataDetector*.

Figure 2 shows the class diagram for this pattern. Some of the classes are the *ContentInspector*, which is intended to check the content of the XML messages sent from/to the applications. The *ContentInspector* consists of an *HarmfulDataDetector* and an *XMLSchemaDetector*. The *HarmfulDataDetector* perform checks for harmful data embedded in the content of the message. The *XMLSchemaValid* checks the validity of the XML documents sent to the application.

3.6 Consequences

The XML Firewall has the following advantages:

- Provides a higher level of security than the Application Firewall because it inspects the complete XML message
- The institution policies to control access are easily defined and administered, as the policies are centralized. This makes the whole system less complex, and thus more secure.
- This facilitates the detection of possible attacks. An Intrusion Detection System can be combined with this firewall. In turn, the IDS can help the firewall block suspicious requests.
- The firewall lends itself to a systematic logging of incoming and outgoing messages.
- As authentication of Clients is performed, it holds regular users responsible of their actions.
- New applications are easily integrated into the system by adding their specific policies.
- New clients can be accommodated by adding new policies to the policy base of an application.

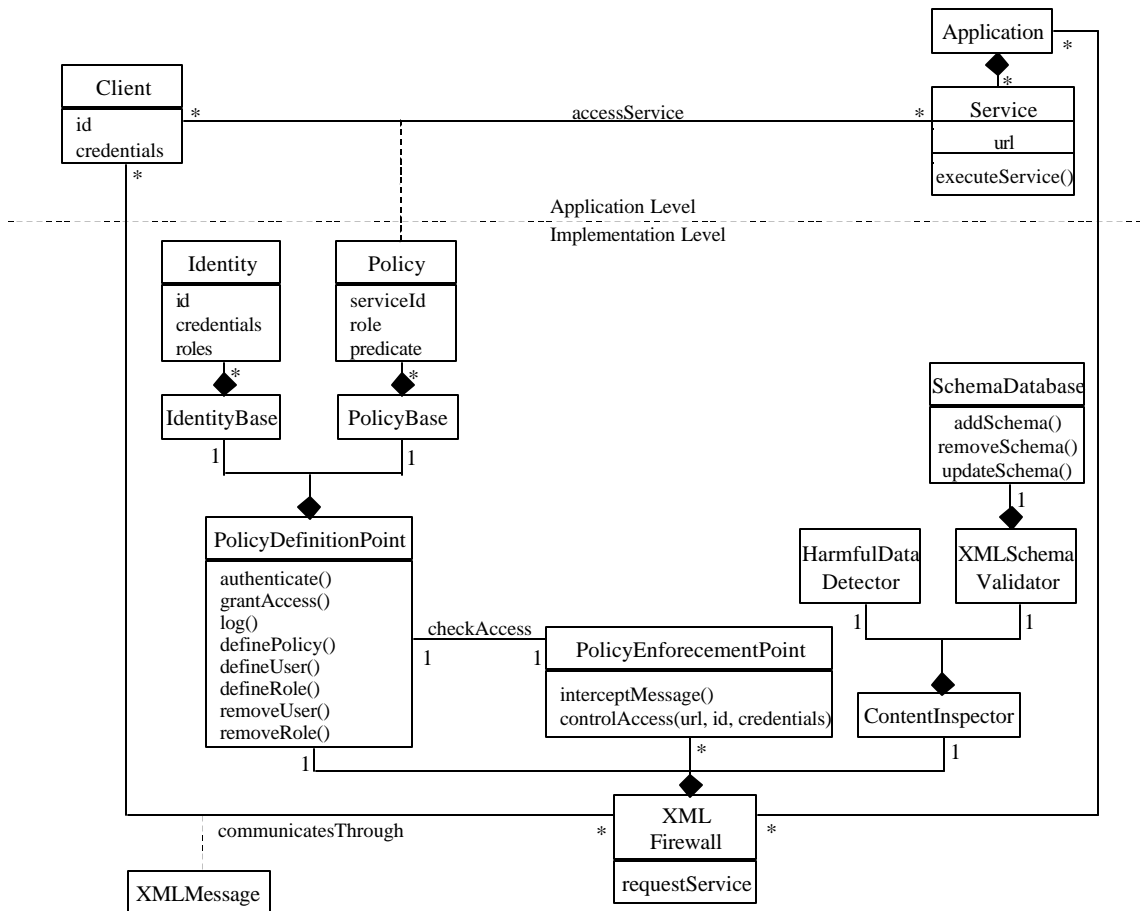


Figure 2. Class diagram for the XML Firewall.

The XML Firewall has the following (possible) liabilities:

- The application could affect the performance of the protected system as it is a bottleneck in the network. This can be improved by considering the firewall a virtual concept and using several firewalls for implementation.
- The solution is intrusive for existing applications that already implement their own access control.
- The application itself must be built in a secure way or normal access to commands could allow attacks through the requests.
- We still need the Operating System to be secure.
- The application could affect the performance of the protected system as it is a bottleneck in the network, and as the XML content checking may create a large overhead.
- The solution is intrusive for existing applications that already implement their own access control.

3.7 Known Uses

This model is used in several commercial products, including Reactivity's XML Firewall [10], Vordel's XML Security Server [12], Westbridge's XML Message Server [13], and Netegrity's TransactionMinder [9].

3.8 Related Patterns

This model is a specialization of the Application Firewall [1]. The Authorization pattern [4] defines the security model for these two firewalls. The Role-Based Access Control pattern, a specialization of the authorization pattern, is applicable if the business policies are respectively defined in terms of roles and rights [4]. This firewall is also a special case of the Single-Point of-Access [14].

4.0 Conclusions

These patterns address two important problems of web services security and they are the beginning of a collection to describe the most important standards [5]. Other possible patterns may consider aspects such as XACML, WS-Sec protocols, and SOAP Cryptographic Extensions.

5.0 References

- [1] N. Delessy-Gassant and E.B.Fernandez, "Patterns for application firewalls", to be published.
- [2] Entegrity AssureAccess, Product Overview. <http://www.entegrity.com/products/aa/aa.shtml>
- [3] E. B. Fernandez and J. C. Hawkins, "Determining role rights from use cases", *Procs. 2nd ACM Workshop on Role-Based Access Control, November 1997*, 121-125.
- [4] E. B. Fernandez and R. Pan. "A Pattern Language for Security Models". In *Proceedings of PLoP*, 2001, http://jerry.cs.uiuc.edu/~plop/plop2001/accepted_submissions/accepted-papers.html
- [5] E.B.Fernandez, "Web services security", chapter in *Web Services Business Strategies and Architectures*, P. Fletcher and M. Waterhouse (Eds.), Expert Press, UK, 2002, 290- 302.
- [6] E. B. Fernandez, M. L. Petrie, N. Seliya, and A. Herzberg. A Pattern Language for Firewalls. In *Proceedings of the PLoP Conference*, 2003.
- [7] E. B. Fernandez, R. Ahmed, and M. Abushadi, The Role-Based Assertion Coordinator pattern", to be published.
- [8] IONA, <http://www.iona.com>
- [9] Netegrity, Inc. A reference architecture. <http://www.netegrity.com/pdfs/refarch.pdf>
- [10] Reactivity. <http://www.reactivity.com>
- [11] SAML, <http://www.saml.org> and <http://www.oasis-open.org/committees/security/>
- [12] Vordel Limited. An in-depth description of Vordel's innovative VordelSecure, the XML security server. http://www.vordel.com/downloads/VS2.1_white_paper.pdf
- [13] Westbridge Technology. <http://www.westbridgetech.com>
- [14] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security". *Procs. PLoP'97 and Chapter 15 in Pattern Languages of Program design, Vol. 4* (N. Harrison, B. Foote, and H. Rohnert, Eds.), Addison-Wesley 2000.