Patterns for Authentication and Authorisation Infrastructures

Roland Erber, Christian Schläger[1], Günther Pernul[2]
Department of Information Systems, University of Regensburg
[1]christian.schlaeger@wiwi.uni-regensburg.de; [2]guenther.pernul@wiwi.uni-regensburg.de

Abstract

In line with the growing success of e-commerce demands for an open infrastructure providing security services are growing stronger. Authentication and Authorisation Infrastructures (AAIs) enhanced with an attribute-based access control model (ABAC) offer such services to service federations and customers. As AAIs are a security enhancing technology, design and implementation must comply with extremely high quality standards. Failures and vulnerabilities in the provided basic security services exponentially affect the service providing processes. Various AAI concepts, frameworks, and products have been developed in the past. Building on these experiences, we define a pattern system for AAIs. It will ensure interoperability and quality of future AAI solutions. The derived pattern system consists of security patterns already published and in use, as well as on open standards like SAML and XACML and related patterns. It can be directly used in the software development cycle, as proposed by different methodologies.

1. Introduction

For e-commerce and distributed computing new demands on infrastructures and service providing have been developed. For Service Providers (SPs) these demands include a higher level of security through fine grained access control (AC) and additional information about customers as well as the possibility to outsource security services to 3rd party providers. Users require better usability with a Single Sign-On (SSO), central maintenance of account data, and the possibility to prove their reputation and trustworthiness. Service providers on the Internet are familiar with infrastructures providing basic security services. Authentication and Authorisation Infrastructures (AAIs) have started with a basic SSO functionality but are nowadays able to manage the complete authorisation and AC process [1].

Over time, developers and software designers have constructed and implemented a variety of AAI architectures. Their knowledge and experience call for a respective pattern system – or in this case a system of security patterns [2]. Patterns can solve specific problems in a given context leaving open the detailed implementation. This is a major advantage when tailoring such open security patterns to the specific user system. [3]

This paper is structured as follows. In section 2 an introduction to AAIs is given. Section 3 introduces a common AAI reference architecture. The preliminaries of the pattern system are given in section 4. The AAI pattern system itself is constructed in section 5. The paper ends with a conclusion.

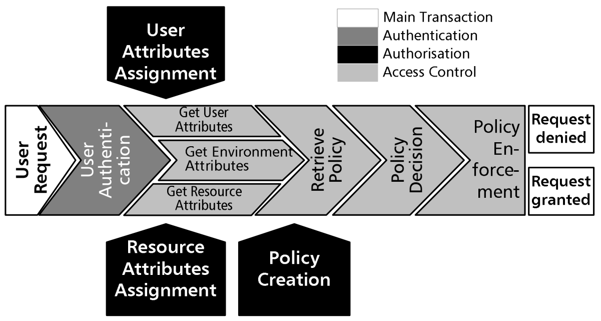

Figure 1. AAI security services

2. Authentication and Authorisation Infrastructures

AAIs make it possible to combine service outsourcing strategies with strengthened security and more flexible and suitable AC techniques. A special benefit lies in the accumulation and exchange of user data over a federation, e.g. user profiles, buying patterns, or earned privileges. Identities can be transferred from one service provider to another making it possible to always use up-to-date address data or proof a good reputation acquired at one federation member.

AAIs as a tool for SPs on the Internet have been discussed on a technical level by [4] in 2004 and in more detail by [1] in 2005. Different architectures, research projects, and products have been analysed and motivation for the stakeholders in such infrastructures has been given. In 2006 [5] defined a process chain of security sub services an AAI is able to perform (Figure 1). This was followed by [6] and existing AAIs were evaluated according to the chain in [5]. The results were a classification and a complete assessment of best practices as well as lessons learned. Using the experiences of GRID AAIs like CAS and VOMS; Privilege Management Infrastructure like PERMIS and AKENTI; and Web Based Solutions like Microsoft's .NET Passport, Liberty's ID-FF, or PAPI, a reference model was derived using exclusively open standards like OASIS's SAML [7] and XACML [8]. The outcome of the analysis is given in Figure 2.

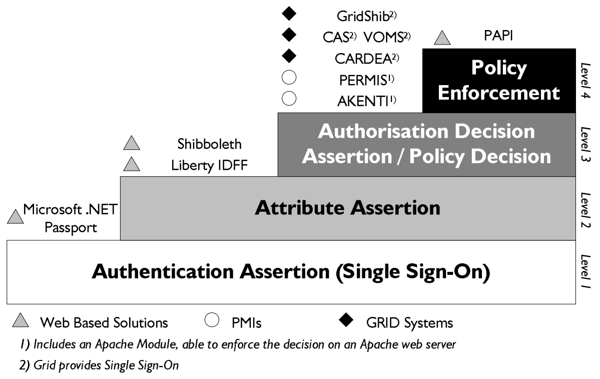
GridShib2)
CAS2) VOMS2)
CARDEA2)
PERMIS1)
AKENTI1)
PAPI
Policy Enforcement
Level 4
Shibboleth
Liberty IDFF
Authorisation Decision Assertion / Policy Decision
Level 3
Microsoft .NET Passport
Attribute Assertion
Level 2
Authentication Assertion (Single Sign-On)
Level 1
Web Based Solutions
PMIs
GRID Systems
1) Includes an Apache Module, able to enforce the decision on an Apache web server
2) Grid provides Single Sign-On

Figure 2. AAI solutions and functionalities (from [6])

3. Common reference model for AAIs

Building on [1, 4, 5] a reference architecture was developed in [6]. Four main design guidelines were realised: a distributed infrastructure, SSO functionalities, attribute-based access control, and open standards. The proposed infrastructure merges AAIs with attribute-based AC. Three entities work together providing services for customers: Service Providers (SP-1…SP-n), Identity Providers (IdP^x), and a centrally maintained trusted AAI server computing an access control decision (PDP).

User α wants access on resource ρ at SP-1. His request is redirected to IdP$^\alpha$ where the user authenticates him as α. For privacy reasons the user's true identity is not revealed to the SP automatically. Therefore the user is identifiable by SP-1 only with an opaque identifier: β. To compute an access control decision IdP$^\alpha$ collects α's attributes from SP-1...n. The compound attributes are forwarded to the PDP together with the access control decision request. The PDP collects the attributes for ρ and the respective environment ε from SP-1. Based on the access policy of SP-1 and the attributes on β, ρ, and ε the decision is derived and forwarded to SP-1. SP-1 enforces the decision locally at his Policy Enforcement Point (PEP). The attributes are transferred between the entities' Policy Information Points (PIP) via SAML tokens. The Policy Administration Point (PAP) manages the different policies in the federation. The according information and data flow is depicted in Figure 3.

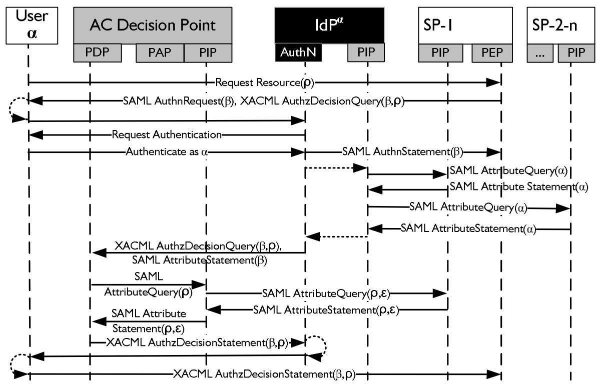
User α
AC Decision Point
PDP
PAP
PIP
IdPα
AuthN
PIP
SP-1
PIP
PEP
SP-2-n
...
PIP
Request Resource(ρ)
SAML AuthnRequest(β), XACML AuthzDecisionQuery(β,ρ)
Request Authentication
Authenticate as α
SAML AuthnStatement(β)
SAML AttributeQuery(α)
SAML Attribute Statement(α)
SAML AttributeQuery(α)
SAML AttributeStatement(α)
XACML AuthzDecisionQuery(β,ρ), SAML AttributeStatement(β)
SAML AttributeQuery(ρ)
SAML AttributeQuery(ρ,ε)
SAML AttributeStatement(ρ,ε)
SAML Attribute Statement(ρ,ε)
XACML AuthzDecisionStatement(β,ρ)
XACML AuthzDecisionStatement(β,ρ)

Figure 3. Reference Model for attribute-based AAI (from [6])

4. Building a pattern system

In accordance with [3], the reference model can be seen as the first step in designing a pattern system for AAIs. Best practices and lessons learned are used. However, [6] extended the mere observation of existing work by open standards. SAML was used for the communication and XACML for the implementation of attribute-based access control. Consequently, this paper is the next step towards the definition of a pattern system for AAIs.

To derive the pattern system the architecture has been analysed according to the occurring data flow. Each functional software component has been specified in Figure 4. Messages and data tokens are listed. The main entities of User, Service Provider Authentication Point AuthN, the XACML components (PEP, PDP, PAP), and Attribute Authority are connected via SAML Requestor and SAML Assertion modules. These two modules substitute the XACML standard's PIP. The combination of SAML and XACML has been shown by [9].

As a result, the SAML Assertion pair is operating as a connector between all AAI components and consequently, all sub-services are exchanging information via SAML Assertion pairs. Using this open interchange format, existing security patterns can be conjoined. For each single security pattern the interface to a SAML request and assertion token needs to be identified.

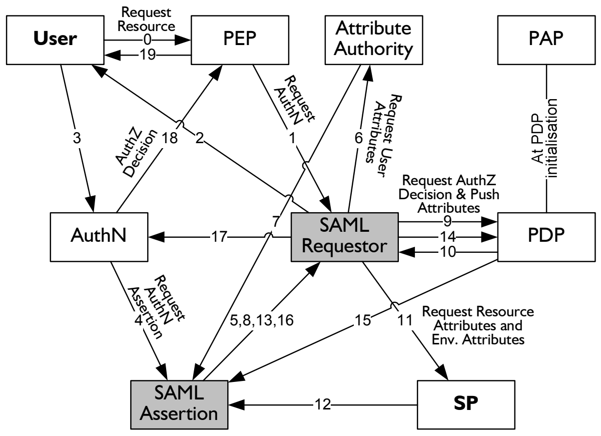

Figure 4. AAI dataflow using SAML entities

5. A Pattern System for AAIs

The pattern system given in Figure 5 is derived from the pattern system presented in [10] and [11] and was adapted for the specific needs of an attribute-based AAI. The patterns are grouped by the AAI components known from the reference architecture. The pattern system has been grouped in four blocks numbered with capital roman numerals (I-IV). Each block consists of further security patterns numbered with lower-case roman numerals (i-iix). In accordance with [3] existing and accepted patterns were used where possible.

5.1 User Authentication Block – I

5.1.1. Identity Federation Pattern – i [11]**.** Serves as a super ordinate pattern of the user authentication component and has got an `<interacts-with>` relationship with the SAML Assertion block. Therefore it is indirectly connected with all other components of the pattern system. It represents an identity federation consisting of several service providers and uses the patterns **Identity Provider**, **Circle of Trust**, **Authenticator Pattern** and **Credential Pattern**. The Identity Federation Pattern allows transporting identity information between the service providers and hence meets the demands for a single sign-on system.

5.1.2. Authenticator Pattern – ii [3]**.** This pattern is an elementary pattern and provides basic authentication mechanisms. The pattern uses the **Credential Pattern** for processing the authentication.

5.1.3. Identity Provider Pattern – iii [11]**.** This pattern supports all IdPs in the AAI. It enables central administration of user identities in a given security domain – for our case, this domain is the AAI federation.

5.1.4 Circle of Trust Pattern – iv [11]**.** Following Liberty's ID-FF idea, a federation consists of various SPs in a so called Circle of Trust. It realises needed trust relations and pseudonymisation.

5.1.5. Credential Pattern – v [12]**.** This is also an elementary pattern. It provides a container for the mediation of authentication and authorisation information in distributed systems. This pattern comes with an `<implemented-As>` relationship, because it is implemented by the **SAML Assertion Pattern**.

5.2 SAML Assertion Block – II

As explained in section 4, SAML Assertions are used as connectors between the single AAI sub-services and security patterns.

5.2.1. Assertion Coordinator Pattern – vi [13]**.** This security pattern represents the central point for the SAML Assertion block and has an `<interacts-with>` relationship with every AAI component. Its major task is to distribute generated SAML Assertions to the participating AAI components. This pattern should be slightly adapted, because in its current form it is applicable to RBAC-based AAIs only. However, the approach by [6] uses ABAC. As has been proven by [14] ABAC is able to subsume other access control models such as DAC, MAC, or RBAC. An adaptation of the original pattern has already been advised in the work of [13].

5.2.2. SAML Assertion - vii [10]**.** vii implements the abstract Credential Pattern (v.) and consequently allocates resources for transporting SAML request or SAML response messages defined in common XML format.

5.2.3 Assertion Builder Pattern – viii [15]. Its task is the generation of SAML Assertion messages in case of a SAML Request or a SAML Response. For this reason it has got a `<creates>` relationship with the SAML Assertion Pattern.

5.3 Access Control Block – III

5.3.1. Attribute-based Access Control – ix [16]. ix represents the basic pattern of the access control component of an attribute-based AAI. The pattern itself provides elementary functionality for attribute-based access control. In the pattern system the actual implementation of this pattern (`<implemented-as>` relationship) follows with the **XACML Access Control Evaluation Pattern**, which is using the open standard XACML [8].

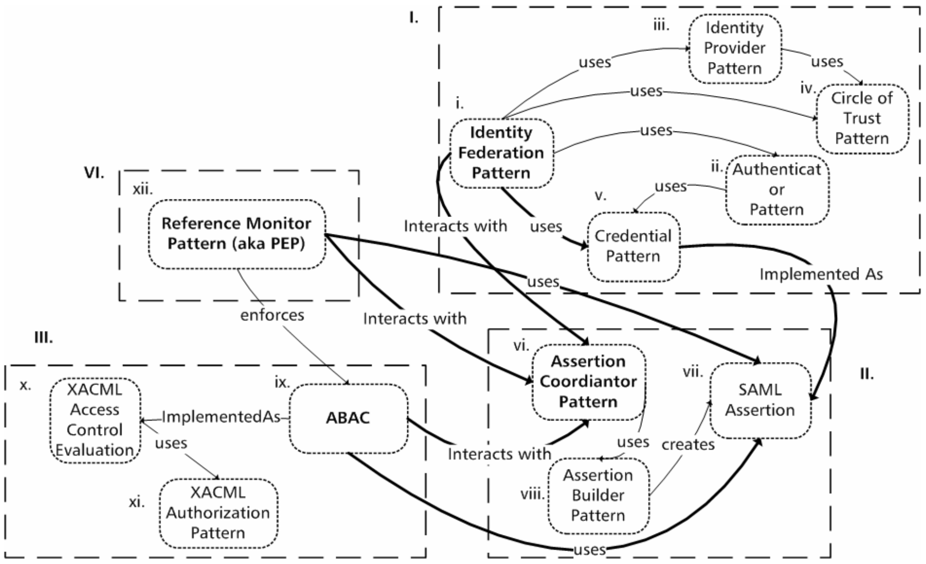

Figure 5. Pattern System for attribute-based AAIs

5.3.2 ***Get Attributes*****,** ***Retrieve Policy*****,** ***Policy Decision***

XACML Access Control Evaluation Pattern – x [17]. x is a concrete implementation of the **Attribute-based Access Control Pattern** and is responsible for collecting attributes, retrieving policies, and calculating access control decisions. This decision is based on the policies that are delivered by the **XACML Authorization Pattern** (`<use>` relationship).

5.3.3. AAI functionality "*Policy Creation*":

XACML Authorization Pattern – xi [17]**.** Covers the policy creation of the process chain and is therefore accountable for creating, deleting, and updating policies, policy sets and rules.

5.4 Policy Enforcement – IV

5.4.1. Reference Monitor Pattern – xii [3]. xii is also known as Policy Enforcement Point (PEP) and responsible for enforcing the access control decision that is calculated by the **XACML Access Control Evaluation Pattern**. This security pattern also has got an `<interacts-with>` relationship with the SAML Assertion block and thus communicates indirectly with the other components of the pattern system. The PEP is the first and the final point of interaction for the user (see Figure 4).

5.5 AAI components not integrated in the pattern system

Two functionalities from Figure 1 have not been integrated in the proposed pattern system: *User Attributes Assignment* and *Resource Attributes Assignment*. These elements of the AAI service security chain are not included, since the attribute assignment for resources and users is not part of the actual access control process.

6. Conclusion

The derived pattern system is able to express completely and exhaustively a generic attribute-based Authentication and Authorisation Infrastructure. Despite the complexity of the given architecture the components can be modularised and each will be assigned a specific and accepted security pattern. Consequently, AAI software designers can now start with the abstract chain of security sub-services as given in the introduction and select the functionality's individual security pattern. This has been done in Figure 6.

With this work two main benefits of security patterns are exploited. Firstly, the potential to learn from other implementation and avoid security flaws and software bugs for business critical software, and secondly, the modularisation of patterns. A change in the AAIs security model, e.g. from ABAC to RBAC, can easily be realised by the respective pattern.

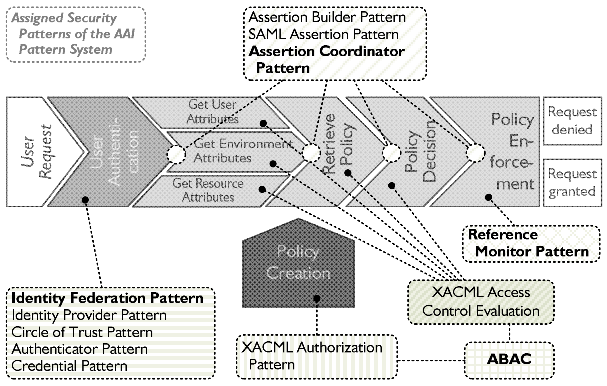
Assigned Security Patterns of the AAI Pattern System
Assertion Builder Pattern
SAML Assertion Pattern
Assertion Coordinator Pattern
User Request
User Authentication
Get User Attributes
Get Environment Attributes
Get Resource Attributes
Retrieve Policy
Policy Decision
Policy Enforcement
Request denied
Request granted
Reference Monitor Pattern
Policy Creation
XACML Access Control Evaluation
Identity Federation Pattern
Identity Provider Pattern
Circle of Trust Pattern
Authenticator Pattern
Credential Pattern
XACML Authorization Pattern
ABAC

Figure 6. AAI services with respective security patterns

7. References

[1] C. Schläger and G. Pernul, "Authentication and Authorisation Infrastructures in b2c E-Commerce", Proc. of the International Conference on E-Commerce and Web Technologies (EC-Web'05), Copenhagen, Denmark, Lecture Notes in Computer Science (LNCS), Vol. 3590, Copenhagen, Denmark, 2005.

[2] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security", in *Pattern Languages of Program Design*, vol. 4, B. F. N. Harrison, and H. Rohnert, Ed. Harlow, England: Addison-Wesley, 2000.

[3] M. Schumacher, E. B. Fernández, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*: John Wiley & Sons, 2006.

[4] J. Lopez, R. Oppliger, and G. Pernul, "Authentication and authorization infrastructures (AAIs): a comparative survey", *Computers & Security*, vol. 23, pp. 578-590, 2004.

[5] C. Schläger, T. Nowey, and J. A. Montenegro, "A reference model for Authentication and Authorisation Infrastructures respecting privacy and flexibility in b2c E-Commerce", Proc. of the 1st International Conference on Availability, Reliability and Security (ARES 2006), Vienna, Austria, Vienna, Austria, 2006.

[6] C. Schläger, M. Sojer, B. Muschall, and G. Pernul, "Attribute-Based Authentication and Authorisation Infrastructures for E-Commerce Providers", Proc. of the International Conference on E-Commerce and Web Technologies (EC-Web'06), Krakow, Poland, Lecture Notes in Computer Science (LNCS), Vol. 4082, Krakow, Poland, 2006.

[7] OASIS Security Services Technical Committee, "Security Assertion Markup Language (SAML)", 2005, online: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security.

[8] OASIS eXtensible Access Control Markup Language Technical Committee, "eXtensible Access Control Markup Language (XACML)", 2005, online: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[9] A. Anderson and H. Lockhart, "SAML 2.0 profile of XACML v2.0 (OASIS Standard)", 2005, online: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf.

[10] E. B. Fernandez, N. Delessy, and M. M. Larrondo-Petrie, "Patterns for Web Services Security", Proc. of the 4th Intl. Workshop on Service-Oriented Architecture and Web Services, part of OOPSLA 2006, Portland, Oregon, USA, 2006.

[11] N. Delessy, E. B. Fernández, and M. M. Larrondo-Petrie, "A Pattern Language for Identity Management", 2nd IEEE Int. Multiconference on Computing in the Global Information Technology (ICCGI 2007), Guadeloupe, French Caribbean, 2007.

[12] P. Morrison and E. B. Fernandez, "The Credential Pattern", Pattern Languages of Programs (PLoP 2006) conference, Portland, Oregon, USA, 2006.

[13] E. B. Fernández, "Two patterns for web services security", Proc. International Symposium on Web Services and Applications (ISWS'04), Las Vegas, NV, Las Vegas, NV, USA, 2004.

[14] T. Priebe, E. B. Fernández, J. I. Mehlau, and G. Pernul, "A Pattern System for Access Control", Proc. of the Annual IFIP WG 11.3 Working Conference on Data and Application Security Sitges, Spain, 2004.

[15] C. Steel, R. Nagappan, and R. Lai, *Core security patterns: best practices and strategies for J2EE, Web services, and identity management*. Upper Saddle River, NJ: Prentice Hall PTR, 2006.

[16] E. B. Fernandez and G. Pernul, "Patterns for Session-Based Access Control", Proc. of the Intl. Conference on Pattern Languages of Programming (PLoP'06), Portland, Oregon, USA, 2006.

[17] N. Delessy and E. B. Fernández, "Patterns for the eXtensible Access Control Markup Language", Proc. of the Pattern Languages of Programs Conference (PloP 2005), Allerton Park, IL, USA, 2005.