

SECURITY DESIGN PATTERNS

Valentin Corneliu PAU, Proffesor, Ph.D.

Marius Iulian MIHAILESCU, Ph.D. candidate

Octavian STANESCU, Inf.

„TITU MAIORESCU” University, Bucharest

ABSTRACT:

Security design patterns have been proposed recently as a tool for the improvement of software security during the architecture and design phases. Since the appearance of this research topic in 1997, several catalogs have emerged, and the security pattern community has produced significant contributions, with many related to design. In this paper, we survey major contributions in the state of the art in the field of security design patterns and assess their quality in the context of an established classification. From our results, we determined a classification of inappropriate pattern qualities. Using a Six Sigma approach, we propose a set of desirable properties that would prevent flaws in new design patterns, as well as a template for expressing them.

1. INTRODUCTION

Good application design is often rooted in appropriate design strategies and leverages proven best practices using design patterns. Design strategies determine which application tactics or design patterns should be used for particular application security scenarios and constraints. *Security Design patterns* are an abstraction of business problems that address a variety of security requirements and provide a solution to the known security related problem(s). They can be architectural patterns that depict how a security problem can be resolved architecturally (or conceptually), or they can be defensive design strategies upon which secure code can later be built.

Core security patterns is a collection of proven design patterns for delivering end-to-end security in J2EE applications, Web services, identity management, and service provisioning. These security patterns differ from traditional infrastructure security design patterns in terms of addressing the end-to-end security requirements of an application by mitigating security risks at the functional and deployment level, securing business objects and data across logical tiers, securing communications, and protecting the application from unauthorized internal and external threats and vulnerabilities.

Typical to Gang-of-four patterns, Core security patterns are structured and represented using a standard pattern template that allows expressing a solution for solving a common or recurring problem. The template captures all the elements of a pattern and describes its motivation, issues, strategies, technology, applicable scenarios, solutions, and examples.

2. SECURITY PATTERN MODEL

To facilitate using the security patterns, we adopted a pattern template that consists of the following:

- *Problem*: Describes the security issues addressed by the pattern.
- *Forces*: Describes the motivations and constraints that affect the security problem. Highlights the reasons for choosing the pattern and provides justification.
- *Solution*: Describes the approach briefly and the associated mechanisms in detail.
 - *Structure*: Describes the basic structure of the solution using UML sequence diagrams and details the participants.
 - *Strategies*: Describes different ways a security pattern may be implemented and deployed.
- *Consequences*: Describes the results of using the security pattern as a safeguard and control measure. It also describes the trade-offs.
- *Security Factors and Risks*: Describes the factors and risks to be considered while applying the pattern.
- *Reality Checks*: Describes a set of review items to identify the feasibility and practicality of the pattern.
- *Related Patterns*: Lists other related patterns from the Security Patterns Catalog or from other related sources.

3. CORE SECURITY PATTERNS CATALOG

The figure 1 illustrates how Core Security Patterns are represented in delivering end-to-end security of a J2EE based application architecture and how it is related in aspects of role and responsibilities in various components and logical tiers - such as *Web Tier*, *Business Tier*, *Web Services Tier*, and *Identity Tier*. In the following sections, we briefly discuss how each pattern is represented specific to its logical tier, how they relate to each other with coexisting component tiers and finally how it contributes to the end-to-end security of an application.

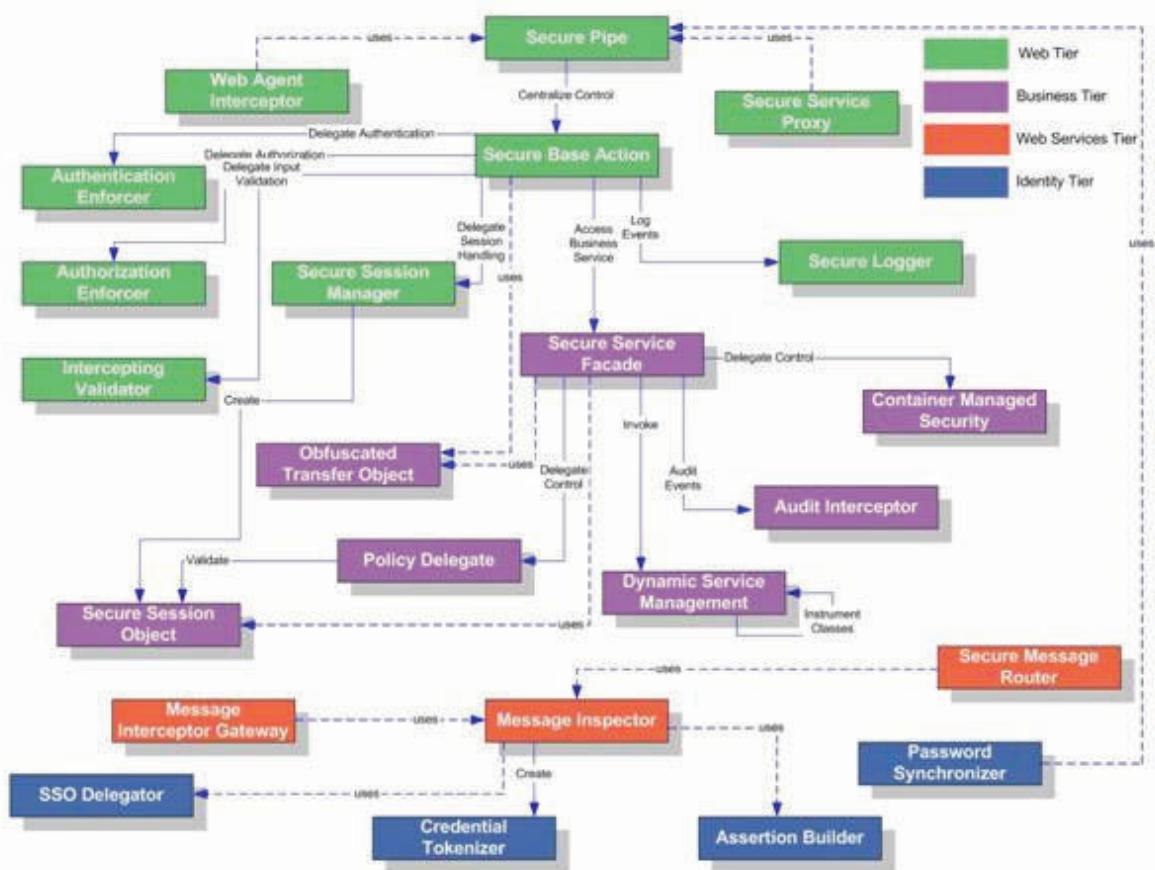


Fig. 1 Representation of Core Security Patterns

4. CLASSIFICATION OF WEB TIER SECURITY PATTERNS

Pattern Name	Standards & Technologies	Description	Related Patterns
Authentication Enforcer	HTTPS; SSL/TLS; IPsec JAAS; JSSE; JCE; JGSS;	This pattern illustrates how a J2EE based application client should authenticate with a J2EE application.	Context Object [CJP]; Intercepting Filter [CJP]
Authorization Enforcer	JACC JAAS; JSSE; JCE; JGSS;	This pattern illustrates how authorization should be enforced after user authentication with a J2EE application.	Context Object; Intercepting Filter [CJP]
Intercepting Validator	JSP Servlets	This pattern refers to secure mechanisms for validating parameters before invoking a transaction. Unchecked parameters may lead to buffer overrun, arbitrary command execution, and SQL injection attacks. The validation of application-specific parameters includes validating business data and characteristics such as data type (string, integer), format, length, range, null-value handling, and verifying for character-set, locale, patterns, context, and legal values.	Message Inspector; Message Interceptor Gateway

Secure Base Action	JSP Servlets	<p>The secure base action is a pattern for centralizing and coordinating security-related tasks within the Presentation Tier. It serves as the primary entry point into the Presentation Tier and should be extended, or used by a Front Controller. It coordinates use of the Authentication Enforcer, Authorization Enforcer, Secure Session Manager, Intercepting Validator, and Secure Logger to ensure cohesive security architecture throughout the Web Tier.</p> <p><i>Refer to Chapter 9, "Securing the Web Tier: Design Strategies and Best Practices," for details.</i></p>	FrontController [CJP]; Command[GoF]; Authentication Enforcer; Authorization Enforcer; Secure Logger; Intercepting Validator
Secure Logger	JMX; Java API for logging	<p>This pattern defines how to capture the application-specific events and exceptions in a secure and reliable manner to support security auditing. It accommodates the different behavioral nature of HTTP servlets, EJBs, SOAP messages, and other middleware events.</p>	Abstract Factory Pattern[GoF]; Secure Pipe;
Secure Pipe	HTTPS; SSL/TLS; IPsec JSSE	<p>This pattern shows how to secure the connection between the client and the server, or between servers when connecting between trading partners. In a complex distributed application environment, there will be a mixture of security requirements and constraints between clients, servers, and any intermediaries. Standardizing the connection between external parties using the same platform and security protection mechanism may not be viable. It adds value by requiring mutual authentication and establishing confidentiality or non-repudiation between trading partners. This is particularly critical for B2B integration using Web services.</p>	Message Interceptor Gateway
Secure Service Proxy	Servlets JAX-RPC SAAJ	<p>This pattern is intended to secure and control access to J2EE components exposed as Web services endpoints. It acts as a security proxy by providing a common interface to the underlying service provider components (for example, session EJBs, servlets, and so forth) and restricting direct access to the actual Web services provider components. The Secure Service Proxy pattern can be implemented as a Servlet or RPC handler for basic authentication of Web services components that do not use message-level security.</p>	Proxy [GoF] Intercepting Web Agent; Secure Message Router; Message Interceptor Gateway; Extract Adapter [Kerievsky]
Secure Session Manager	Servlets EJB	<p>This pattern defines how to create a secure session by capturing session information. Use this in conjunction with Secure Pipe. This pattern describes the actions required to build a secure session between the client and the server, or</p>	Context Object [CJP]

		between the servers. It includes the creation of session information in the HTTP or stateful EJB sessions and how to protect the sensitive business transaction information during the session.	
Intercepting Web Agent	JMX Web server plugin	This pattern helps protecting Web based J2EE applications through a Web Agent that intercepts requests at the Web Container and provides authentication, authorization, encryption, and auditing capabilities.	Proxy [GoF]

Table 1. Web Tier Design Patterns Classification

5. BUSINESS TIER SECURITY PATTERNS

Pattern Name	Standards & Technologies	Description	Related Patterns
Audit Interceptor	Java API for Logging; Log4J	Works in conjunction with the Secure Logger pattern provides instrumentation of the logging aspects in the front, and the Audit Interceptor pattern enables the administration and manages the logging and audit in the back-end.	Secure Logger Intercepting Filter [CJP]
Container Managed Security	EJB	This pattern describes when and how to declare security-related information for EJBs in a deployment descriptor.	Secure Pipe
Dynamic Service Management	JMX	This pattern provides dynamically adjustable instrumentation of security components for monitoring and active management of business objects.	Secure Pipe; Secure Message Router
Obfuscated Transfer Object	JCE	This pattern describes ways of protecting business data represented in transfer objects and passed within and between logical tiers.	Transfer Object [CJP];
Policy Delegate	JACC EJB XACML	This pattern creates, manages, and administers security management policies governing how EJB tier objects are accessed and routed.	Secure Base Action; Business Delegate [CJP]
Secure Service Facade	EJB	This pattern provides a session façade that can contain and centralize complex interactions between business components under a secure session. It provides dynamic and declarative security to back-end business objects in the service façade. It shields off foreign entities from performing illegal or unauthorized service invocation directly under a secure session. Session information can be also captured and tracked in conjunction with the Secure Logger pattern. <i>Refer to Chapter 10, “Securing the Business Tier – Design Strategies and Best Practices,” for details.</i>	Secure Service Proxy; Session Façade [CJP]

Secure Session Object	EJB	<p>This pattern defines ways to secure session information in EJBs facilitating distributed access and seamless propagation of security context.</p> <p><i>Refer to Chapter 10, “Securing the Business Tier – Design Strategies and Best Practices,” for details.</i></p>	Transfer Object [CJP]; Session Façade[CJP]
------------------------------	-----	---	---

Table 2. Business Tier Security Patterns

6. WEB SERVICES TIER SECURITY PATTERNS

Pattern Name	Standards & Technologies	Description	Related Patterns
Message Inspector	XML Encryption; XML Signature; SAAJ; JAX-RPC; WS-Security; SAML; XKMS;	This pattern checks for and verifies the quality of XML message-level security mechanisms, such as XML Signature and XML Encryption in conjunction with a security token. The Message Inspector pattern also helps in verifying and validating applied security mechanisms in a SOAP message when processed by multiple intermediaries (actors). It supports a variety of signature formats and encryption technologies used by these intermediaries.	Message Interceptor Gateway, Secure Message Router
Message Interceptor Gateway	JAX-RPC; SAAJ; WS-Security XML Signature; XML Encryption; SAML XACML WS-*	This pattern provides a single entry point and allows centralization of security enforcement for incoming and outgoing messages. The security tasks include creating, modifying, and administering security policies for sending and receiving SOAP messages. It helps to apply transport-level and message-level security mechanisms required for securely communicating with a Web services endpoint.	Secure Access Point, Message Inspector, Secure Message Router
Secure Message Router	WSS-SMS XML Signature XML Encryption WS-Security Liberty Alliance SAML XKMS	This pattern facilitates secure XML communication with multiple partner endpoints that adopt message-level security and identity-federation mechanisms. It acts as a security intermediary component that applies message-level security mechanisms to deliver messages to multiple recipients where the intended recipient would be able to access only the required portion of the message and remaining message fragments are made confidential.	Secure Access Point, Message Inspector, Message Interceptor Gateway

Table 3. Web Services Tier Security Patterns Classification

7. IDENTITY MANAGEMENT AND SERVICE PROVISIONING

Pattern Name	Standards & Technologies	Description	Related Patterns
Assertion Builder	SAML; Liberty Alliance	This pattern defines how an identity assertion (for example, authentication assertion or authorization assertion) can be built.	Single Sign-on Delegator

Credential Tokenizer	SAML; Liberty Alliance	This pattern describes how a principal's security token can be encapsulated, embedded in a SOAP message, routed, and processed. .	Secure Session Object
Single Sign-on (SSO) Delegator	SAML; Liberty Alliance	This pattern describes how to construct a delegator agent for handling a legacy system for single sign-on (SSO).	Service Locator [CJP] Business Delegate [CJP]
Password Synchronizer	SPML	This pattern describes how to securely synchronize principals across multiple applications using service provisioning.	

Table 4. Identity Management and Service Provisioning Classification

8. CONCLUSIONS

In this paper, we reported a survey on the state of the art of security design patterns and found important undesirable properties in them. Basing ourselves on those results, and using a Six-Sigma approach, we were able to determine the desirable properties of security design patterns as an engineering product. Furthermore, we proposed a new template for pattern description that would allow to better specify patterns in the future.

In future research, we aim at defining new patterns for representing, managing and implementing security policies, as well as providing their usage-specific instances.

9. BIBLIOGRAPHY

- [1] ISO/IEC 10181-1:1996, *Security frameworks for open systems: Overview*, 1996.
- [2] ISO/IEC 10181-2:1996, *Security frameworks for open systems: Authentication framework*, 1996.
- [3] ISO/IEC 10181-3:1996, *Security frameworks for open systems: Access control framework*, 1996.
- [4] C. Alexander, S. Ishikawa, and M. Silverstein, *A pattern language*, Oxford University Press, 1977.
- [5] B. Blakley and C. Heath, *Security design patterns*, Tech. Report G031, Open Group, 2004.
- [6] A. Braga, C. Rubira, and R. Dahab, *Tropyc: A pattern language for cryptographic software*, 1998.
- [7] E. B. Fernandez and R. Pan, *A pattern language for security models*, *Proceedings of the PLoP 2001*, 2001.
- [8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: Elements of reusable object-oriented software*, Addison-Wesley, 1994.
- [9] J. R. Hauser and D. Clausing, *The house of quality*, *IEEE Engineering Management Review* **24,1** (1996), pp. 24–32.
- [10] F.L. Brown Jr. and E. B. Fernandez, *The authenticator pattern*, *Proceedings of the PLoP 99*, 1999.
- [11] D. M. Kienzle and M. C. Edler, *Final technical report: Security patterns for web application development*, Tech. Report DARPA Contract # F30602-01-C-0164, 2002.
- [12] D. M. Kienzle, M. C. Elder, D. Tyree, and J. Edwards-Hewitt, *Security patterns repository*, 2002.
- [13] T. Priebe, E.B. Fernandez, J.I. Mehlau, and G. Pernull, *A pattern system for access control*, *Proceedings 18th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2004.
- [14] S. Romanosky, *Security design patterns part 1*, 2001.
- [15] J. Yoder and J. Barcalow, *Architectural patterns for enabling application security*, *Proceedings of the PLoP 97*, 1997.