Security By Default: Patterns-driven Security Design for Java Enterprise Applications

Ramesh Nagappan CISSP

NEJUG Meeting – June 8, 2006
Burlington, MA
Setting Expectations

What you can take away!

• We'll take a look at a "Proactive approach" for building trustworthy Java Enterprise applications
  > Based on the work - “Core Security Patterns”

• We'll discuss about a “Prescriptive guidance” for integrating security during the software development lifecycle.
  > It doesn't compel you to secure every component without making a case.
  > If you are building a security sensitive application – Then it is definitely a consideration.
About Me

- **Ramesh Nagappan**
  - Java Architect @ Sun Microsystems
  - Co-author of “Core Security Patterns” and few other books/articles.
  - CISSP
  - Ph.D dropout
  - Contributor to several open-source projects and XML Web services specifications
  - Extensive experience with Java Security, Java Enterprise applications, XML Web services and Identity Assurance technologies.
    - Closely involved with projects related to Government, Law Enforcement, Intelligence, Defense, Financial and Healthcare organizations.
    - Current technology focus is on strong authentication using PKI, Smartcards and Biometrics.
  - Personal email: nramesh@post.harvard.edu
Presentation Outline

• The State of IT Security
  > Myths and Realities
  > Critical Security Flaws and Vulnerabilities

• Security-By-Default
  > Motivation and Strategies

• Introducing Secure UP - Methodology
  > Security Wheel – Realizing Security requirements
  > Secure UP – Security Design Methodology

• Introducing Security Patterns
  > Security Patterns Template
  > Security Patterns Catalog

• Patterns-Driven Security Design: A Web Services Case study
  > Identifying and applying Security Patterns
  > Web Services Security - Best Practices Considerations

• Q & A
The State of IT Security
The State of IT Network Landscape
Fading boundaries, growing pervasive

- Enterprise
- Drop-In Center
- Customer Location
- Leveraged Site
- Train
- Home
- Campus
- Beach
- Airport
- Region Hub
- Partner Location
- Pool
- Hotel
- Flexible Field Office
- www.coresecuritypatterns.com
The State of IT Security - Myths
Many Perspectives

- Username/Passwords
- Roles
- Groups
- LDAP

- Programming Rules
- Constraints
- Cryptography
- Permissions

- Security Infrastructure Products

- Firewall
- Router
- ACLs
- IDS

- Company wide Security Policy

IT Management
Network Administrator
System Administrator
Application Developer
Technology Vendor
End User

* Cartoon Based on ancient Asian Indian fable
The State of IT Security - Realities

• There is NO ROLLBACK for an application security breach.  
  > Reactive security and port-mortem fixes are very limited.

• Security is often considered as an ADD-ON or after-thought solution during software development life-cycle (SDLC).
  > Use-cases does not capture security requirements.
  > Security is often considered as Non-functional requirement not Business enabler.
  > Security considerations are realized only during deployment phase.

• Traditional architecture and design process fails to analyze for security risks and trade-offs.
  > Results inefficient system architecture vulnerable to exploits and attacks.
  > Compelling need for a repeatable and reusable security processes and tools.

• Convergence of legacy applications and enabling of SOA standards based on-demand solutions introduces myriad of newer security challenges.
  > There is no ONE SECURITY SOLUTION that fits all.
• **Identity frauds** are the emerging fastest growing crime in the world.
  > Identity theft, Identity frauds and Impersonation crimes are increasing day-by-day.
  > Application infrastructure is not flexible to incorporate stronger authentication options.

• **Adopting to proprietary security solutions with compatibility issues.**
  > Integration and Interoperability NIGHTMARE leads to extensibility dead-end.
  > Bridging proprietary security products often results complex disjointed solutions.
  > Business software is changing faster than software security technologies.

• **Lack of Management priorities.**
  > Security practices are often determined at the IT department level and then directed to the management.
  > Bottom-up approach often ends-up with lesser funds and resources.

• **Lack of expertise** with identifying and mitigating application-level risks and vulnerabilities
  > Network and System administrators tend to ignore content-level attacks such as malicious code injection, cross-site scripting, XML attacks and so on.
The State of IT Security - Realities continued..

- **Adopting to inefficient infrastructure and operational controls.**
  - Choosing infrastructure with known security flaws prone to platform specific threats.
  - Lack of audit control over operations leads to failure in detecting malicious activities.
  - Lack of operational controls with business applications affects regulatory compliance.

- **Heightened Security and Privacy awareness everywhere**
  - Since 9/11, Security has taken an unprecedented importance in all industry segments.
  - Everyone is keen on knowing about their security options.
  - Organizations are pushed to manage their own risks and liabilities.
  - Sarbanes-oxley, HIPAA, GLBA, HSPD-12

- **Growing need for boundary-less commerce without compromising security**
  - Organizations are moving from brick-and-mortar to network-centric IT enterprises.
  - Network boundaries are slowly fading! On-demand services getting its importance.
  - Increasing need for agility and ubiquitous access to information.
Development Puzzles

• How do we proactively identify risks and vulnerabilities?
  > Does it help build my 100% Security goal or joke!!

• How do we implement safeguards and countermeasures?
  > Is there a magic or silver bullet available?
  > How do we build application-level defense at all logical tiers?

• How do we prevent identity related crimes?
  > How do we establish access control policies
  > How do we enforce authentication and authorization that prevents identity frauds.

• How do we assess and test the safeguards and countermeasures?
  > Does it guarantee our security as threat free and future-proof?

• How do we identify and resist internal and external attacks?
  > Network attacks, Intentional and unintentional application abuses, Disgruntled employee sabotages!!
Development Puzzles continued..

• How to obfuscate Java code?
  > How to protect the executable from reverse engineering?

• How to enable single sign-on and global logout?
  > How to authenticate and authorize users in a Portal?
  > How to prevent session hijacking?

• How to choose and apply cryptographic mechanisms?
  > What is the cost in terms of performance?

• How to provide secure access from devices?

• How to establish secure communication in a workflow?
  > How to manage the limitations of SSL?

• How to fortify XML Web services based applications?
  > How to prevent XML content-level attacks and vulnerabilities?
  > How to choose and apply XML security standards?
  > How to verify for Interoperability?
Critical Network/OS Flaws and Exploits

- Man-in-the-Middle / Sniffing
- TCP SYN/ACK attacks
- IP Spoofing
- PING of Death – SMURF attack
- Denial of Service / Distributed DOS attack
- Dictionary attack / Brute force attacks

- Virus attack
- Birthday attack
- Audit and logging failures
- Suspicious router advertisement
- Trojan horse/Zombie/Back door
- Missing Patch - Vulnerability
- Hardening/Minimization issues
- Rootkits
Critical Application Flaws and Exploits

- Input validation
- Output sanitation
- Buffer overflow
- Data injection
- Improper error handling
- Weak session identifiers
- Weak Password exploits
- Session theft
- Cross-site scripting

- Insecure configuration data
- Broken authentication/ access control
- Audit and logging failures
- Improper error handling
- Denial of Service
- Man-in-the-middle
- Multiple sign-on issues
- Weak Encryption
- Coding / Deployment problems
Critical Web Services Security Exploits

Web Services is not an exception! Infact, it introduces a lot newer challenges.

- XML Denial of Service
- Message Replay / Web Services ping-of-death attack
- Man-in-the-middle / Message Interception
- Message Injection/Manipulation
- WSDL Disclosure / UDDI attacks
- Message Validation abuses

- XML Schema Poisioning
- Message Routing attack
- Auditing and Logging
- Weak Security Tokens
- Compliance issues
- Single Sign-on Provider issues
- Malicious Attachments
The Identity Crisis?

- Identity is often realized based on username/password based authentication and authorization.
  - Passwords are highly vulnerable as it is easier to steal or share or forgotten or used without the consent of the owner.
- How can I trust you?
  - How do we establish access control policies
  - How do we enforce authentication and authorization that prevents identity frauds.
- How can I do single sign-on and global logout?
- How can we federate the Identity in a B2B environment?
- How do we securely provision user accounts in multiple resources?
The Impact of Web 2.0?

- The notion of using Web as a Platform.
  - With the power of scripting languages and light-weight programming model to deliver rich internet clients.
  - AJAX, PHP, Python, Ruby
  - DWR (Direct Web Remoting) allows to run code in Web browser to execute Java functions on a Web server.
- Common security issues with AJAX client-side scripts
  - Input Validation
  - Parameter Tampering
  - Session maintenance issues
  - SSL Overhead due to multiple asynchronous calls.
  - Server-side application methods exposure
The End-users demand?

- **Confidentiality**
  > Only individuals who are privileged have access to information.

- **Integrity**
  > The assurance that information remains correct at all times and has not been altered or destroyed.

- **Availability**
  > Those who are privileged have access to information are able to get to it in a timely manner.
Security By Default
Security-By-Default
A notion that identifies security at all OSI layers for known risks

- Application Security
- Transport Security
- Network Security
- Physical Security

OSI 7-layers
The IT Ecosystem

End User Client Services (e.g., Portals, Web Services Access)

Content Delivery Services / Presentation

Value Added Services / Composite Applications / Workflow

Business Application Execution Containers

Policy  Trust  Identity  AuthN  AuthZ  Privacy  Integrity  Audit

Directory (LDAP)  ERP  Middleware  Database (SQL)  Identity/Access

Network Systems w/ Operating System and Storage

www.coresecuritypatterns.com
Security-By-Default: Motivation

Security is implemented using an HOLISTIC APPROACH and LAYERED DEFENSE STRATEGY that identifies and mitigates for all known risks and vulnerabilities during design or refactoring.
Security-By-Default: Core Requirements

- Network Perimeter Security / Intrusion Detection
- Hardened / Minimized OS
- Auditing and Traceability
- Integrity
- Confidentiality
- Non-repudiation
- Availability / Service continuity
- Monitoring and Manageability

- Authentication
- Authorization
- PKI
- Tamper-proof Logging
- Classification & Labeling
- Identity and Access Management
- Identity Provisioning
- Compliance
- Interoperability at
Security-By-Default: Strategies

• Define a Process and Methodology for Security Design
  > Allows to integrate security as part of SDLC – From requirements till deployment.
  > Allows to incorporate process for gathering security requirements, identify risks, mitigate them and perform trade-offs.

• Adopt Patterns-driven and Best practices based security architecture and design.
  > Helps to incorporate reusable safeguard solution for known risks and vulnerabilities.
  > Helps to proactively mitigate risks traditionally identified using prototyping or testing.

• Introduce Reality Checks to review applied security principles and its behavior before deployment.
  > Assists in the analysis of whether the applied design principles are practicable.

• Knowledge/No-Knowledge based Security Testing and Auditing.

• Adopting Defensive Strategies – Proactive & Reactive actions for known security breaches.
Introducing
Secure UP - Methodology
Security Wheel
Realizing Security requirements using Hub and Spokes

IT Ecosystem

Perimeter Security
Identity Management
Configuration
Hardened OS
Service

www.coresecuritypatterns.com
©2009 Core Security Patterns
Introducing Secure UP

- Based on Unified Process (RUP)
- Prescribes a Security Discipline to bake-in security from ground-up
- Applies a layered defense strategy realizing “Security Wheel”
- Mitigates potential security risks and vulnerabilities at the early design phase
- Adopts a patterns-driven security architecture
- Enforces best practices and reality checks
Key Secure UP Activities and Artifacts

- **Risk Analysis and Mitigation**
  - Identifying the threats and assessing the possible damage to justify safeguards.
  - Cost benefit Analysis
  - Annual Rate of occurrence (ARO) and Annual Loss Expectancy (ALE).
  - Identify residual risks

- **Trade-off Analysis**
  - Weighing Pros and Cons of a safeguard
  - Identifying and comparing alternative safeguards for rational decisions.

- **Policy Design**
  - Defining rules and practices that regulates protection of an application.

- **Factor Analysis**
  - Identify security factors for each infrastructure component (Application provider, client devices, single sign-on support etc)
Key Secure UP Activities and Artifacts

• Tier Analysis
  > Review the factors that impact security mechanisms of different tiers

• Threat Profiling
  > Profiling the architecture and application configuration for potential security weaknesses.
  > A Threat Profile may identify and list threats and vulnerabilities.

• Black box and White box testing
  > Zero Knowledge or Black box testing (Penetration testing).
  > Full Knowledge or White box testing for known weakness and tolerance levels.

• Service Continuity and Disaster Recovery
  > Determine the tolerance level for potential threats
  > Define fault detection and exception handling process
  > Define high availability and fault tolerance
Secure UP - Activities

Define Business Security Requirements → Create Conceptual Security Model → Define Candidate Security Architecture → Risk Analysis → Trade-off Analysis → Identify Security Patterns

Define Security Requirements → Candidate Security Architecture → Risk Analysis → Trade-off Analysis → Identify Security Patterns → Security Design

Perform Risk & Trade-off Analysis → Security Design → System Test → Implementation → Unit Test

Validate Testing & Auditing → Implement Prototype → Deploy → Security Monitoring → Security Audits

White Box Tester → Black Box Tester

White Box Testing → Found Hole → Passed

Black Box Testing → Found Hole → Passed

Operations Staff → Security Auditor

Deploy → Security Monitoring → Security Audits
Driving the Security Design Process

1. Define Security Requirements
2. Identify Security Patterns and Create Security Design
3. Perform Risk & Trade-off Analysis
4. Candidate Security Architecture
5. Implement Prototype
6. Validation Testing & Auditing
Secure UP – Process Steps

1. Define Security Requirements
2. Define Candidate Architecture
3. Risk Analysis & Mitigation
4. Trade-off Analysis
5. Policy Design
6. Factor Analysis
7. Tier-Analysis
8. Threat Profiling
9. Security Design & Implementation
10. Black-box and White-box testing
11. Define Service continuity and disaster recovery
12. Security Monitoring and Auditing
Introducing Security Patterns
Security Pattern Template

• Problem
  > Describes the security issues addressed by the pattern

• Forces
  > Describes the motivations and constraints that affect the security problem - Reasons for choosing the pattern.

• Solution

• Structure

• Strategies
  > Describes the different ways a security pattern may be implemented or deployed.

• Consequences
  > Results of using the pattern as a safeguard or countermeasure.

• Security factors and risks

• Reality checks
  > Review items to identify feasibility and practicality of using the pattern.
Patterns Categorization

Rationale

- Categorized based on **Logical Tiers** of the network-centric application or services.

  - Web / Presentation Tier
  - Business / Application Logic Tier
  - Web Services / Resource Integration Tier
  - Identity / Access Control Tier
Security Patterns and its Relationships
Patterns-driven Security Design
Example Scenario
A Healthcare Scenario

Let’s consider a common Healthcare provider scenario involving:

- Hospitals
- Doctors
- Pharmacies
- Laboratories
- Insurance providers

What are the business problems related to security?

Assuming XML Web Services based Integration environment – How security should be addressed?
Business Problems related to Security

Business Communication scenarios

• Insurance Provider ↔ Patient info ↔ Health provider
• Physician ↔ Patient info ↔ Pharmacy
• Health provider ↔ Patient info ↔ Laboratories
• Health provider ↔ Patient info ↔ Insurance provider
• Pharmacy ↔ Patient info ↔ Insurance provider

Key Security Challenges

• Message integrity and confidentiality
• Message Verification and Validation
• Message Authentication, Authorization
• Message logging and auditing
• Message/Element level security
• Message routing to multiple endpoints
• Message origin verification
• Message Compliance and interoperability
Known Security Risks

**Interruption**: Possible attack on availability

**Interception**: Possible attack on confidentiality

**Modification**: Possible attack on integrity

**Fabrication**: Possible attack on authenticity
Key Security Requirements

- Authentication
- Authorization
- Auditing and Traceability
- Message Integrity
- Message Confidentiality
- Non-repudiation
- Availability and Service continuity
- Identity and Policy
- Security Interoperability
Core Web Services Security Essentials

Message Layer Security

- XML/SOAP/WSDL
- WS-Security
- XML Digital Signature
- XML Encryption
- SAML
- REL

Transport Layer Security

- HTTP over SSL/TLS
- TCP
- IP

Network Layer Security

Application
Presentation
Session
Transport
Network
Data link
Physical
Identified Security Patterns

- Message Interceptor Gateway
- Message Inspector
- Secure Message Router
- **Assertion Builder**
- Secure Logger
- Audit Interceptor
- Obfuscated Transfer Object
Message Interceptor Gateway

Problem
You want to use a single entry point and centralize security enforcement for all incoming and outgoing messages.

Forces
- You want to block and prevent all direct access to the exposed service endpoints.
- You want to intercept all XML traffic and inspect the complete XML message/attachments.
- You want to verify the message integrity and confidentiality for eavesdropping and tampering.
- You want to enforce transport-layer security using Two-way SSL/TLS (Mutual authentication)
- Protect the exposed WSDL descriptions from public access and revealing operations.
- Apply filter mechanisms for content, payload size and message representation.
- Monitor and identify replay and DOS attacks by tracking and verifying the IP addresses, hostnames, message timestamps and other message sender specific information.
Message Interceptor Gateway

Continued

Solution

- Message Interceptor Gateway pattern is a proxy to the Web services infrastructure and provides a centralized entry point encapsulating access to all target service endpoints.
- It works as a security enforcement point.
- It secures the incoming and outgoing XML traffic by securing the communication channels.

Strategies

- XML Firewall Strategy
- Web Proxy infrastructure Strategy
Message Inspector

Problem
You want to verify and validate the quality of message-level security mechanisms applied to XML Web services.

Forces

• You want to examine message-level security for structure and content, verifying their uniqueness, confidentiality, integrity and validity.

• You want to identify the applied security-tokens and assertions representing the identity and policies.

• You want to monitor and identify XML based DOS and REPLAY attacks by tracking and verifying security tokens, signatures, message correlation, message expiry or timestamps.

• You want to verify messages for interoperability and standards compliance.

• You want to enforce a centralized logging based on the security actions and decisions

• You want reusable API mechanisms for managing and processing the message-level security.
Message Inspector

Continued

Solution

• Message Inspector is a modular or a pluggable component that integrates with infrastructure components.

• Executes a chain of tasks as a preprocessing or post processing intermediary for all incoming and outgoing messages.

• It works as a security decision point

Strategies

• Message Handler strategy

• Identity provider agent strategy
Secure Message Router

Problem

You want to securely communicate with multiple partner endpoints applying message-level security and identity federation mechanisms.

Forces

- You need a security intermediary for Web services based workflow or multiple service endpoints.
- You need to configure element-level security and access control.
- You want to revealing only the required portions of a protected message to a target recipient.
- You want to enable SSO (Ex: generating SAML assertions and XACML based ACLs) and global logout notification.
- You want to send notification of identity registration and termination.
- You want to dynamically apply security criterion through message transformations, canonicalizations before forwarding it to its intended recipient.
Secure Message Router

Continued

Solution

- Secure Message Router is security intermediary infrastructure that aggregates access to multiple endpoints.

- Acts on the incoming messages and dynamically applies security logic for routing messages to multiple end-point destinations.

- Makes use of a security configuration utility to apply end-point specific security decisions and mechanisms.

Strategies

- Messaging Provider strategy
- Liberty SSO strategy
Assertion Builder

Problem

You need a structured and consistent approach to gather security information (for example, SAML assertions) about the authentication action performed on a subject, attribute information about the object, or an authorization request from a trusted service provider.

Forces

• You want to avoid duplicate program logic for building authentication assertions, authorization decision assertion and attribute statements.

• You want the flexibility for handling client requests for SAML assertions from a Servlet, EJB or a JAX-RPC/SOAP client.

• You need to abstract all processing and control logic for creating SAML assertion statements.
Assertion Builder
Continued...
Secure Logger

Problem

You require all requests and responses must be securely logged for security auditing and debugging purposes.

Forces

- You need to log sensitive information that should not be accessible to unauthorized users.
- You need to ensure the integrity of the data logged.
- You want to capture output at one level for normal operations, but at all levels prior to an exception.
- You want to centralize control of logging in the system.
Secure Logger
Continued

Strategies

• Secure Data Logger Strategy
• Secure Log Store Strategy
Audit Interceptor

Problem

You want to intercept and audit requests and responses to and from the business tier.

Forces

- You want centralized, declarative auditing of service requests.
- You want perform auditing of service de-coupled from the services themselves.
- You want implement pre-process and post-process audit handling of service requests, including auditing of exceptions.
Audit Interceptor

Continued

Strategies

- Intercepting Session Facade Strategy
Obfuscated Transfer Object

Problem

You need a way to protect critical value objects as it is passed within application and across its logical tiers.

Forces

- You want to protect sensitive data passed in transfer objects from being captured in console messages, log files or even audit logs.
- You want the transfer object to be responsible for protecting the data in order to reduce code and also prevent business components from inadvertently exposing sensitive data.
- You want to specify which data elements are protected, since not all data should be protected and may need to be exposed.
Obfuscated Transfer Object

Strategies

- Masked List Strategy
- Sealed Object Strategy
Best Practices in Web Services Security

1. Choose and implement standards-based security infrastructure components.
2. Protect your network perimeter with firewalls, router ACLs and IDS.
3. Enforce communication security ensuring confidentiality and integrity.
4. Choose an XML-aware security infrastructure at the perimeter level
5. Restrict all direct access to service endpoints, WSDL and XML Schemas
6. Verify and validate all messages before processing
7. Inspect messages for all content-level threats and vulnerabilities
8. Use timestamps and correlation to determine the validity of a message
9. Apply Message element-level encryption and signature mechanisms
10. Adopt secure logging, monitoring and auditing mechanisms
11. Adopt XML-security aware appliances for performance
12. Perform service penetration tests and hardening of environment before deployment

...and more
Further Reading *(Shameless Plug)*

- **Core Security Patterns: Best Practices and Strategies for J2EE, Web Services and Identity Management**

- Visit [CoreSecurityPatterns.Com](http://CoreSecurityPatterns.Com) for more information and resources.

- Feedback to authors@coresecuritypatterns.com
Q & A
Thank You

Ramesh Nagappan  CISSP
nramesh@post.harvard.edu