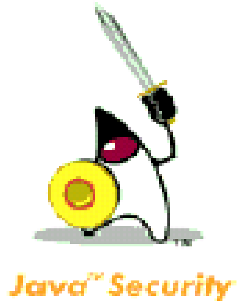
Security By Default: Patterns-driven Security Design for J2EE and Web Services

Ramesh Nagappan CISSP
Christopher Steel, CISSP

INFOSEC Conference, New York – June 28, 2006



core SECURITY PATTERNS

www.coresecuritypatterns.com

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 does'nt compel you to secure every component without making a case.
  - > If you are building a security sensitive application – Then it is definitely a consideration.

Presentation Outline

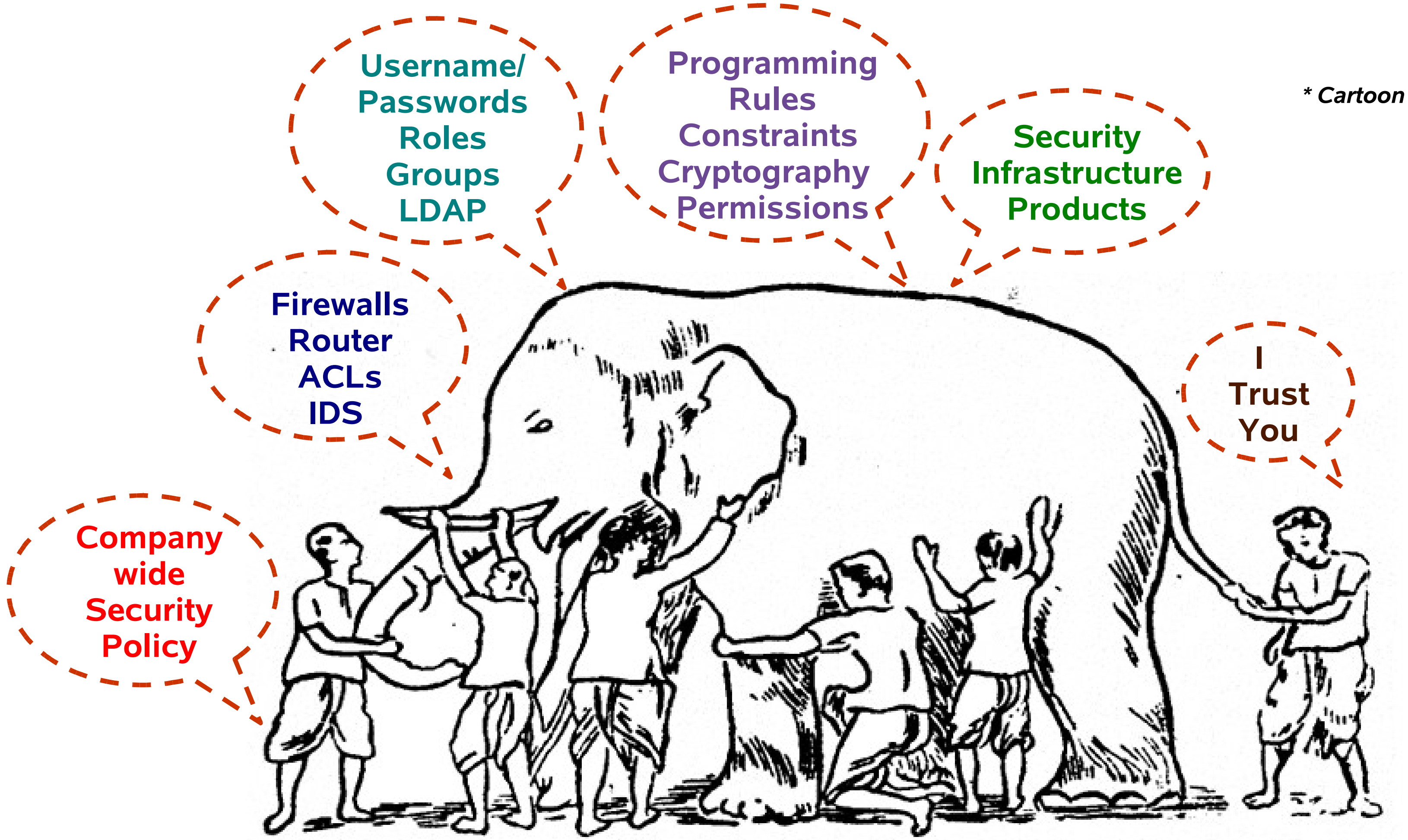
- The State of IT Security
  - Myths and Realities
  - Critical Security Flaws and Vulnerabilities
- Security-By-Default
  - Motivation and Strategies
- Introducing Secure UP - Methodology
  - Security Wheel – Realizing Security requirements
  - Secure UP – Security Design Methodology
- Introducing Security 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 Security - Myths

Many Perspectives

** Cartoon Based on ancient Asian Indian fable*


Username/ Passwords Roles Groups LDAP
Programming Rules Constraints Cryptography Permissions
Security Infrastructure Products
Firewalls Router ACLs IDS
I Trust You
Company wide Security Policy

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

The State of IT Security - Realities

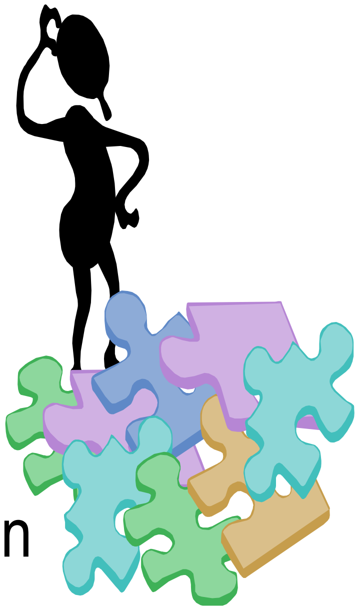
- Security is often considered as an ADD-ON or after-thought solution during software development life-cycle (SDLC).
  - *Use-cases does not capture security requirements.*
  - *Security is often considered as a Non-functional requirement not Business enabler.*
  - *Security considerations are realized 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.*
- There is NO ROLLBACK for an application security breach.
  - Reactive security and port-mortem fixes are very limited.

The State of IT Security - Realities continued..

- 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.

Security Development Puzzles

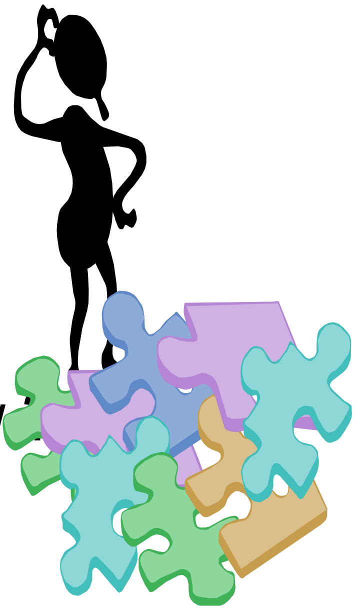
- How do I proactively identify risks and vulnerabilities ?
  - > *Does it help build my 100% Security goal or joke !!*
- How do I implement safeguards and countermeasures ?
  - > *Is there is a magic or silver bullet available ?*
  - > *How do I build application-level defense at all logical tiers ?*
- How do I prevent identity related crimes ?
  - > *How do I establish three-factors of authentication.*
  - > *How do I enforce stronger authentication that prevents identity frauds.*
- How do I assess and test the safeguards and countermeasures?
  - > Does it guarantee my security as threat free and future-proof ?
- How do I identify and resist internal and external attacks on my application ?
  - > Network attacks, Intentional and unintentional application abuses, Disgruntled employee sabotages, Phishing, Man-in-the-middle attacks !!



Security Development Puzzles

continued..

- How can I obfuscate my Java/.NET code ?
  - > *How to protect the executable from reverse engineering ?*
- How can I 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 can I provide secure access from mobile and PDA devices?
- How can I 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 Flaws and Known Exploits

Common Network, Host, Transport, OS and Application level attacks

Data/SQL injection

Input Validation issue

Man-in-the-middle

Firewall issues

Cross-site scripting

Insecure configuration Data

Broken Authentication

Error Handling

IP Spoofing

Dictionary/Password guessing

Buffer overflow

Ping-of-Death

Denial of Service Attack

Missing Patch

Birthday attack

Broken Authorization

Weak Session ID

Hardening/Minimization issue

Deployment issues

Phishing and Spoofing

SMURF attack

Weak encryption

Weak Passwords

Multiple sign-on

TCP SYN attack

Session Hijacking/Theft

Non-repudiation issues

Output sanitation

Rootkit

Auditing/Logging issue

Missing Router ACL

. . . plus a growing list of unknown flaws and exploits

Critical Web Services Flaws & Exploits

Web services is not an exception…..it adds newer challenges

Multiple recipient routing

WSDL Disclosure

Message Validation Abuses

XML Parser overload

XML Denial of Service

Single sign-on provider issue

Message Replay Attack

Weak Security Tokens

Web Services - Ping-of-Death

Standards Compliance

Public Registry Attack

Message Injection and Manipulation

Man-in-the-middle

Malicious Attachments

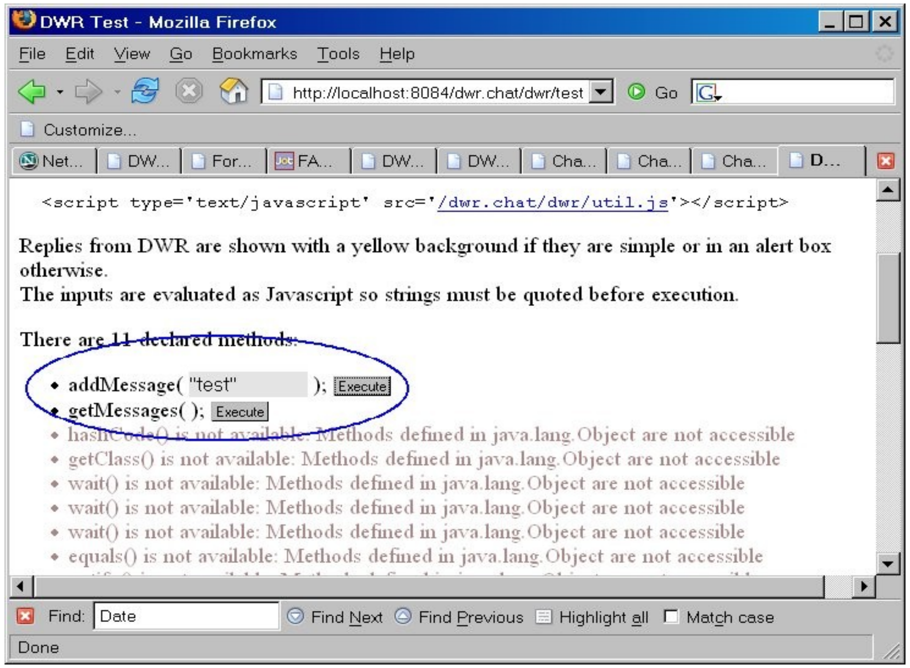
The Identity Crisis ?

- 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 ?

"Do you have another card? This one's been reported stolen."

The Impact of Web 2.0 ?

- The notion of using Web as a Platform.
  - > *Using light-weight scripting 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.*
  - > *XML over HTTP requests*
- 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*


DWR Test - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8084/dwr.chat/dwr/test
Go
Customize...
Net... DW... For... FA... DW... DW... Cha... Cha... Cha... D...

```
<script type='text/javascript' src='/dwr.chat/dwr/util.js'></script>
```

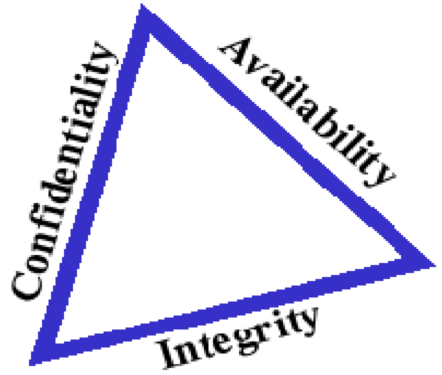
Replies from DWR are shown with a yellow background if they are simple or in an alert box otherwise.
The inputs are evaluated as Javascript so strings must be quoted before execution.

There are 11 declared methods:

- addMessage("test"); Execute
- getMessages( ); Execute
- hashCode() is not available: Methods defined in java.lang.Object are not accessible
- getClass() is not available: Methods defined in java.lang.Object are not accessible
- wait() is not available: Methods defined in java.lang.Object are not accessible
- wait() is not available: Methods defined in java.lang.Object are not accessible
- wait() is not available: Methods defined in java.lang.Object are not accessible
- equals() is not available: Methods defined in java.lang.Object are not accessible

DWR Testkit

The End-users demand ?



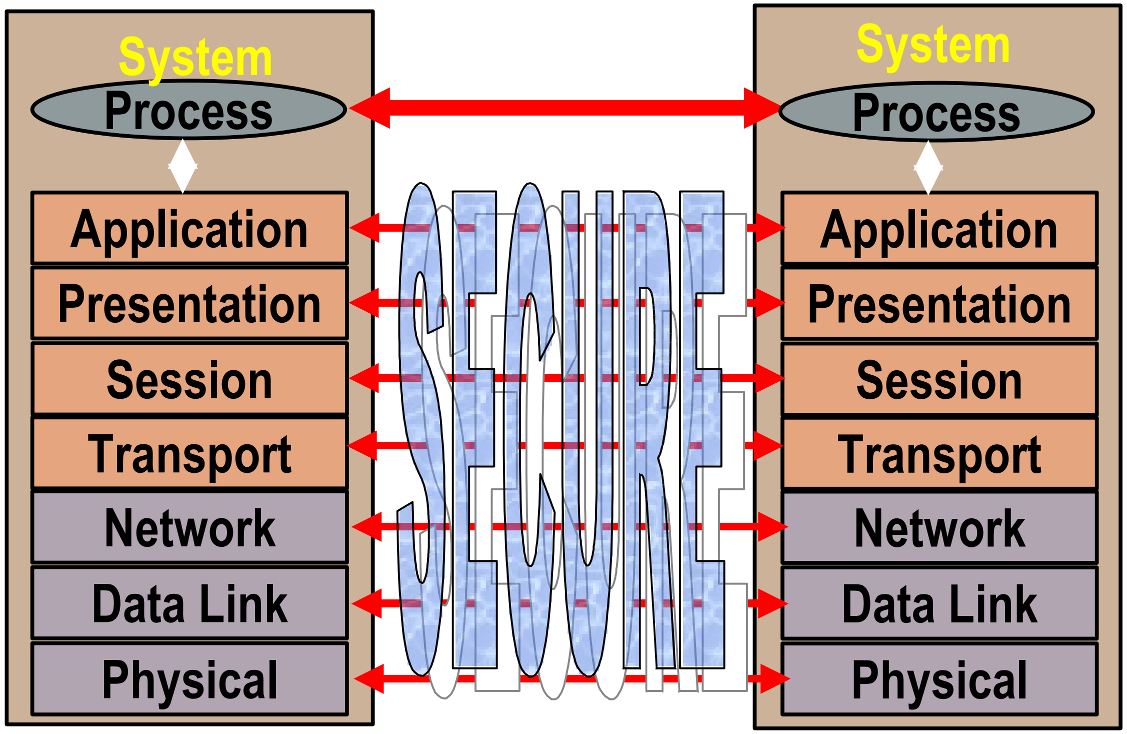
- **Confidentiality**
  - > Only individuals who are privileged have access to information.
- **Integrity**
  - > The assurance that information remains correct at all times and has not been altered or destroyed.
- **Availability**
  - > Those who are privileged have access to information are able to 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

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

Security-By-Default: Strategies

- Define a Process and Methodology for Security Design
  - *integrate security as part of SDLC – From requirements till deployment.*
  - *Gathering security requirements, identify risks, mitigate them and perform trade-offs.*
- Adopt Patterns-driven and Best practices based security architecture and design.
  - *Incorporate reusable safeguard solution for known risks and vulnerabilities .*
  - *Proactively mitigate risks traditionally identified using prototyping or testing.*
- *Introduce Reality Checks to review applied security principles and its behavior before deployment.*
  - *Analyse for whether the applied design principles are practicable.*
- *Knowledge/No-Knowledge based Security Testing and Auditing.*
- *Adopt Defensive Strategies – Proactive & Reactive actions for known security breaches.*

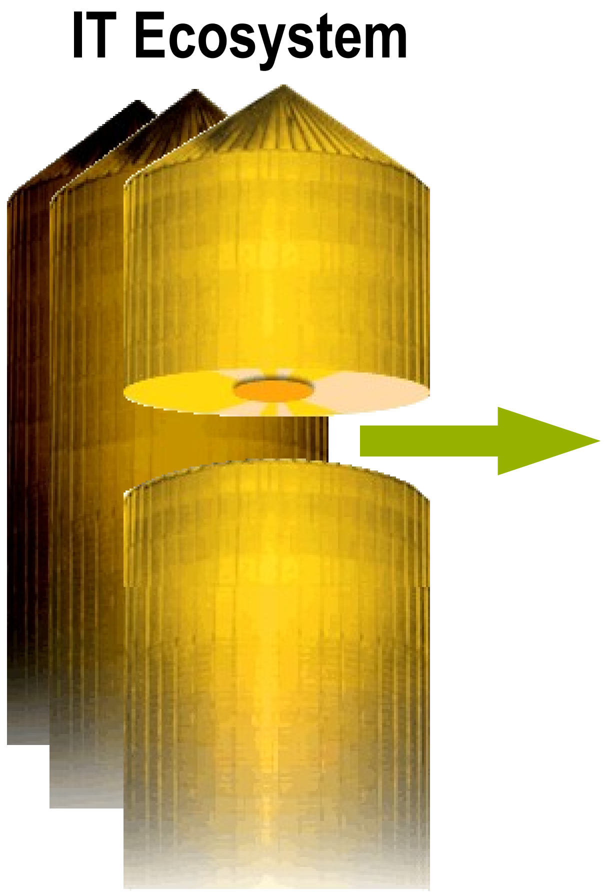
Security-By-Default: In Principle

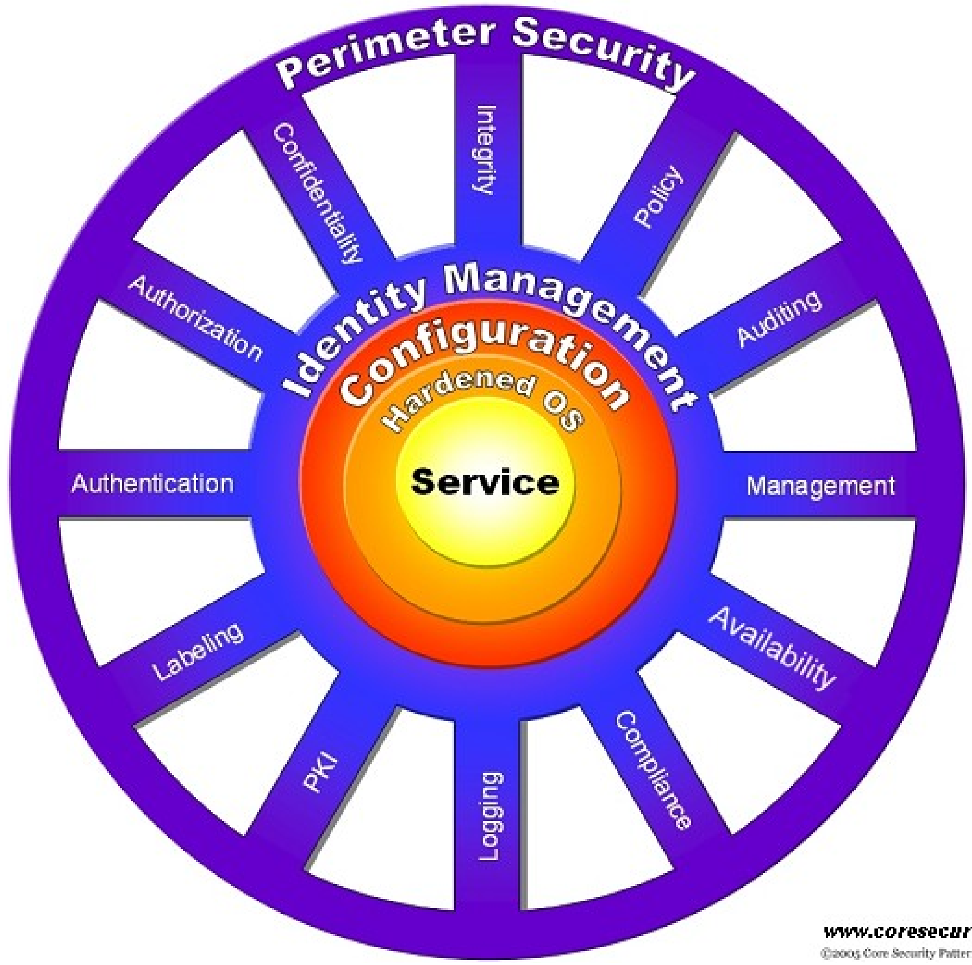
Hack It vs **Use of Patterns & Best Practices**

Introducing

Secure UP - Methodology

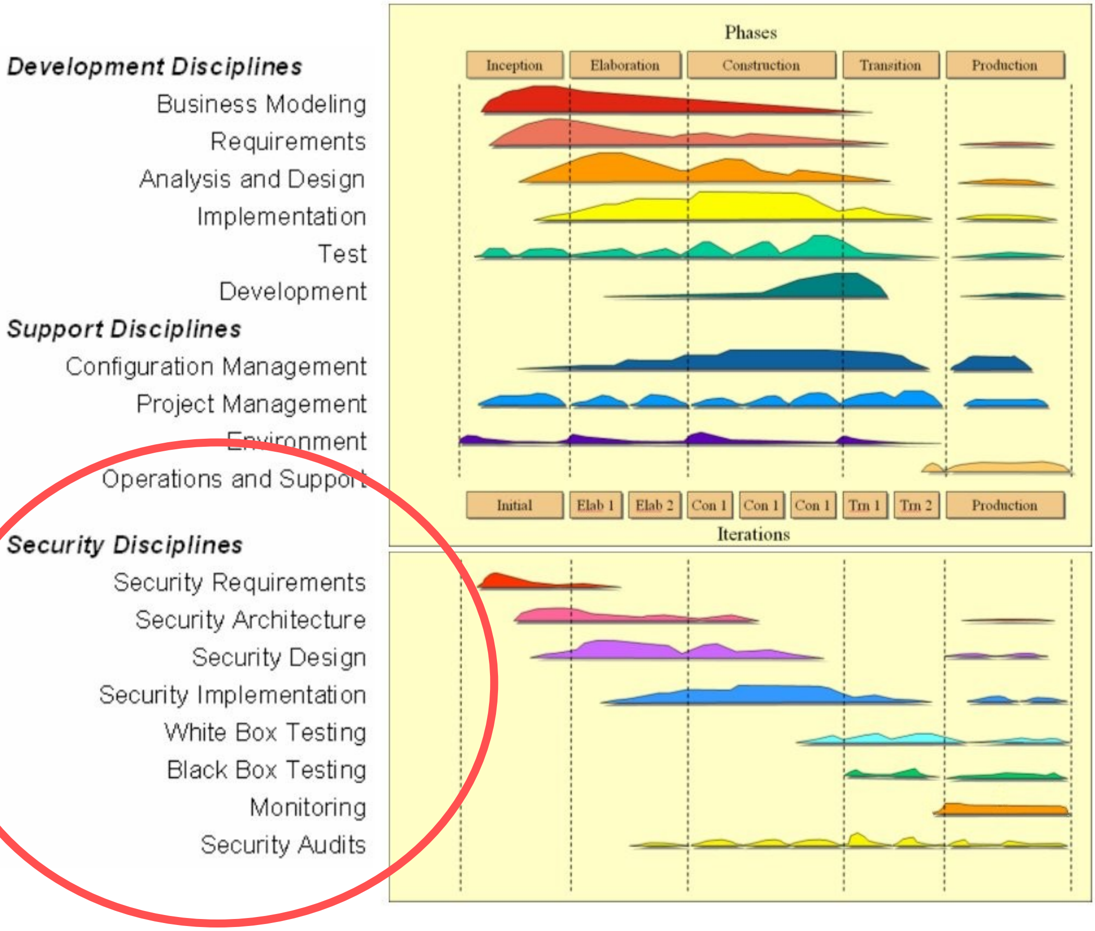
Security Wheel
Realizing Security requirements using Hub and Spokes



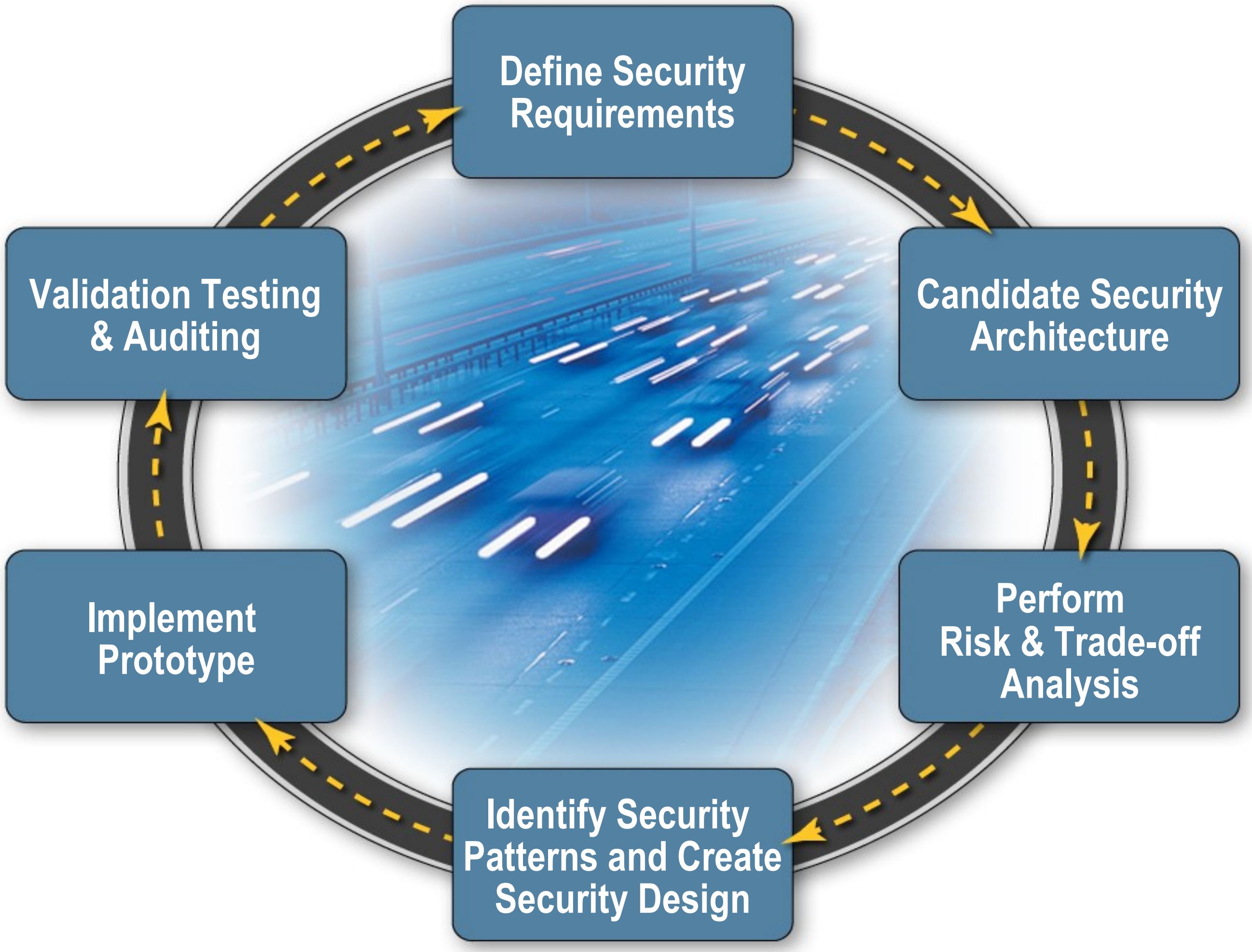


Introducing Secure UP

- 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



Driving the Security Design Process


Define Security Requirements
Candidate Security Architecture
Perform Risk & Trade-off Analysis
Identify Security Patterns and Create Security Design
Implement Prototype
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

Key Secure UP Activities and Artifacts

- Risk Analysis and Mitigation
  - > Identify the threats and assess the possible damage
  - > justify safeguards and countermeasures
  - > Cost benefit Analysis
- Trade-off Analysis
  - > Weigh the Pros and Cons of a safeguard
  - > Identify and compare alternative safeguards for rational decisions.
- Policy Design
  - > Define 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)

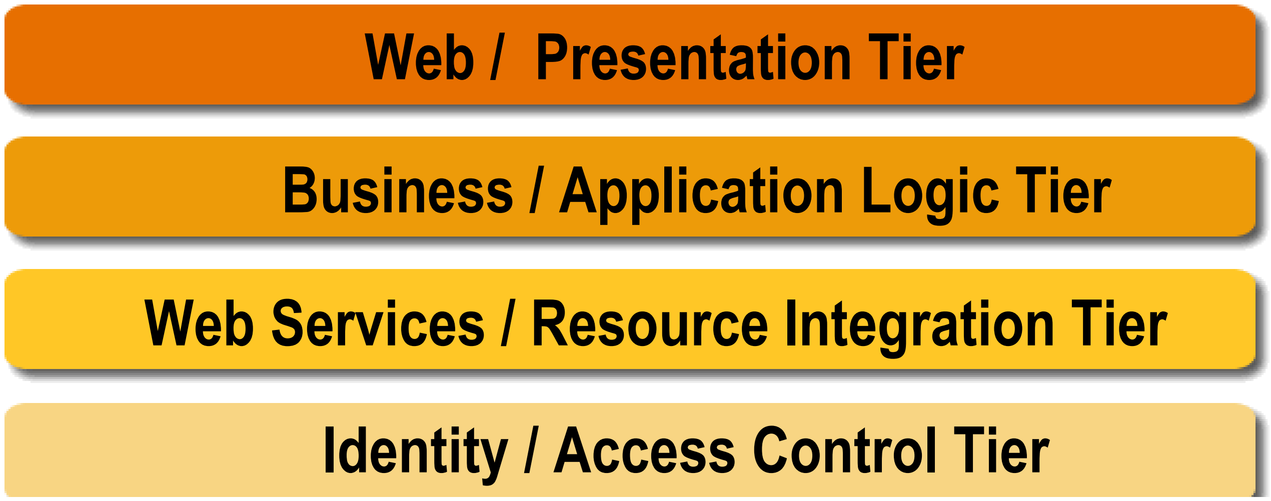
Key Secure UP Activities and Artifacts

- 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

Introducing Security Patterns

Security Patterns
Rationale & Categorization

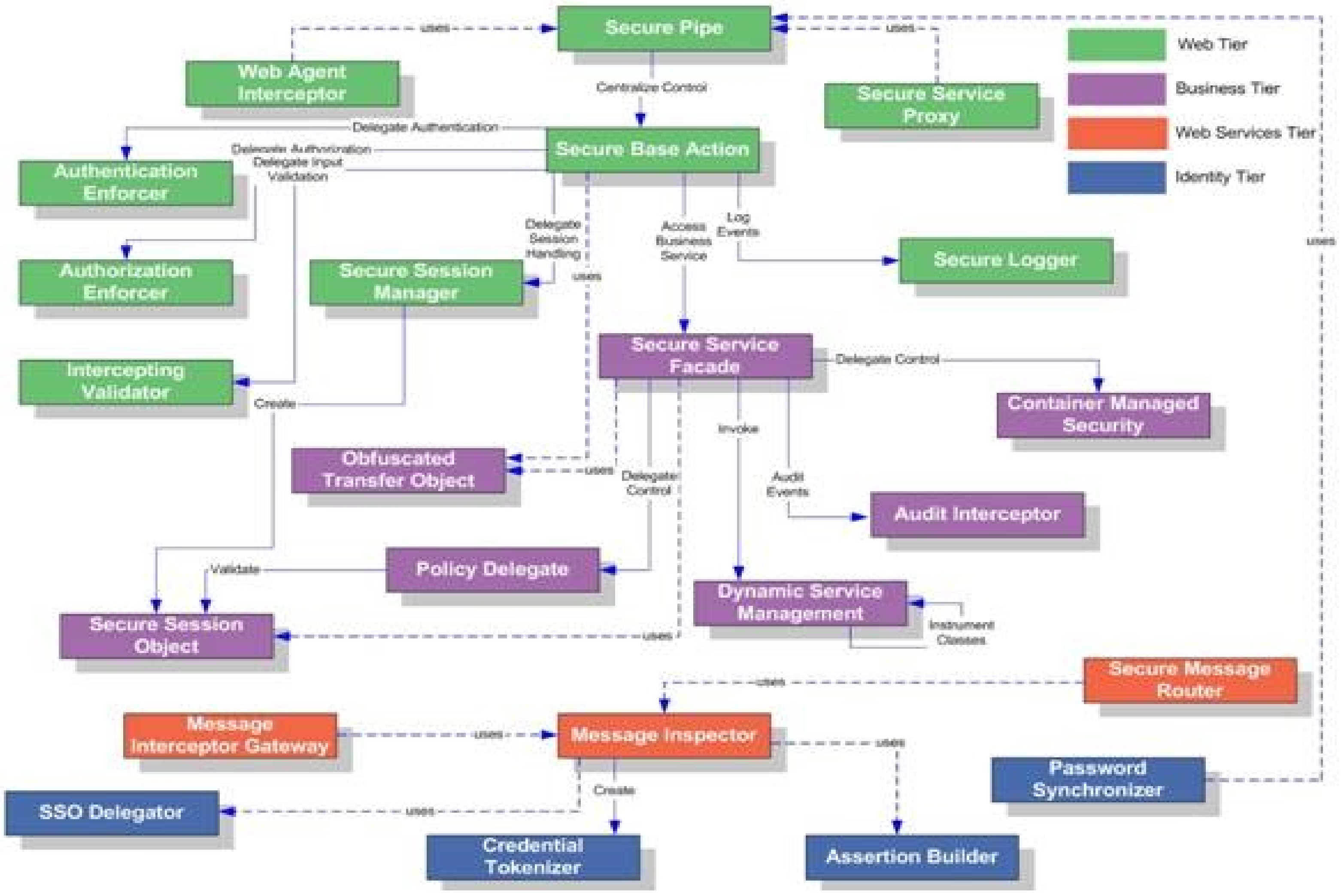
- Pattern: Proven Solution to a recurring security design problem
  - > Represent best practices
  - > Common Vocabulary
- Categorized based on **Logical Tiers** of the network-centric application or services.


Web / Presentation Tier
Business / Application Logic Tier
Web Services / Resource Integration Tier
Identity / Access Control Tier

Security 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.

Security Patterns and its Relationships



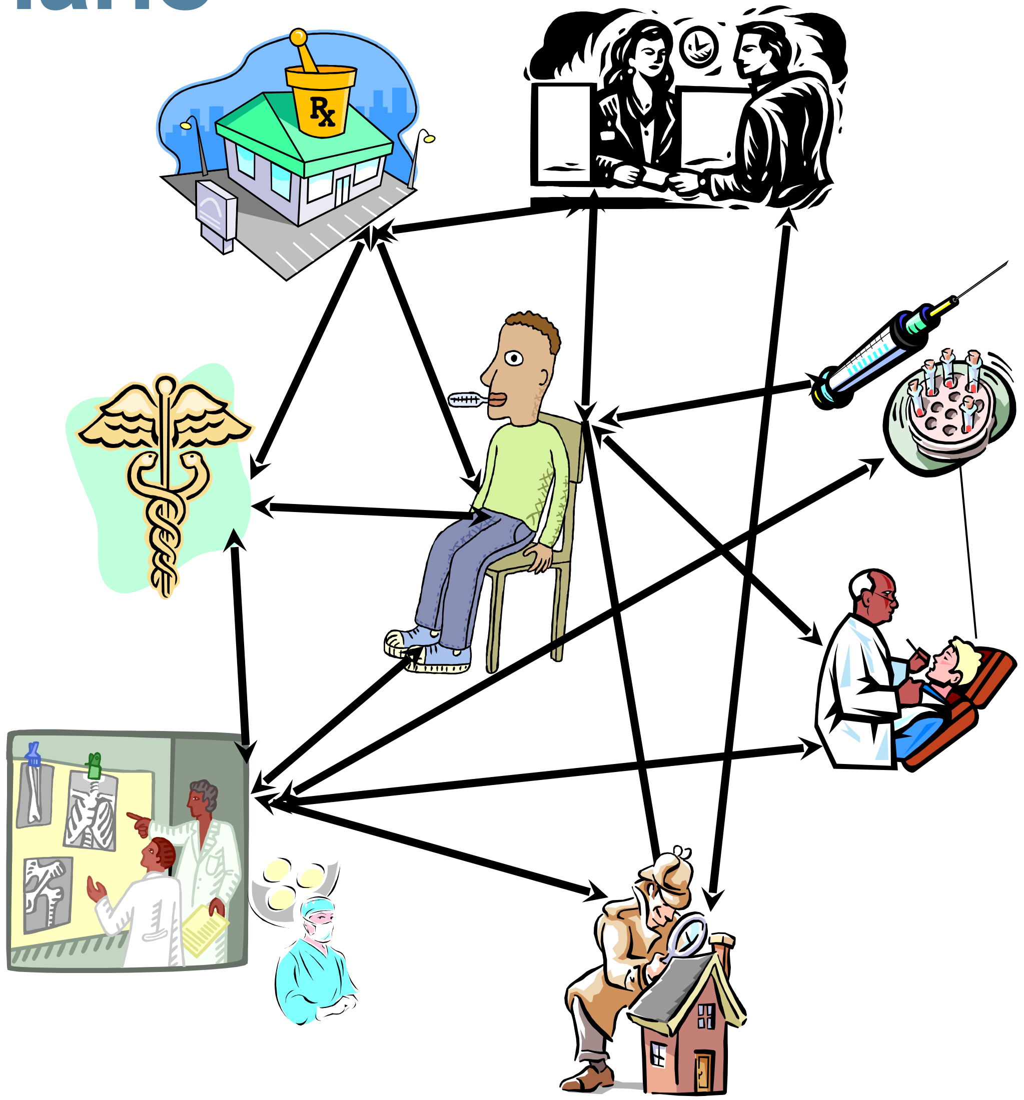
Patterns-driven Security Design Example Case Study

A Healthcare Scenario

Let's consider a Healthcare Service Provider scenario involving:

- Hospitals
- Doctors
- Pharmacies
- Laboratories
- Insurance providers
- Financial Institutions
- Medicare/Medicaid

Assumes a candidate architecture using XML Web Services and Identity federation using an Identity Provider.



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

MANDATORY

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

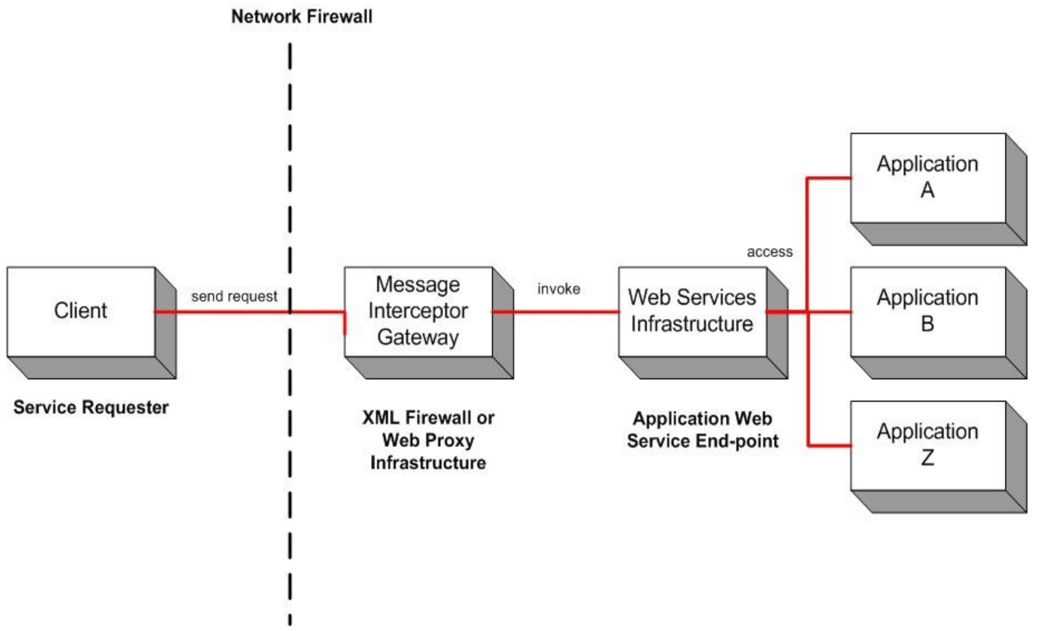
- You want to block and prevent all direct access to the exposed service endpoints.
- You want to Intercept all XML traffic and inspect the complete XML message/attachments.
- You want to verify the message integrity and confidentiality for eavesdropping and tampering.
- You want to enforce transport-layer 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.


Network Firewall
Client
send request
Message Interceptor Gateway
invoke
Web Services Infrastructure
access
Application A
Application B
Application Z
Service Requester
XML Firewall or Web Proxy Infrastructure
Application Web Service End-point

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.

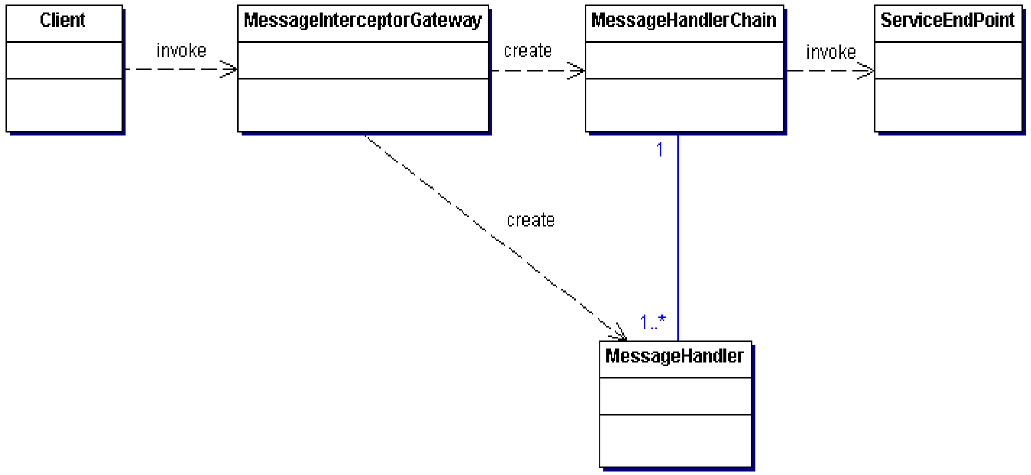
Forces

- 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

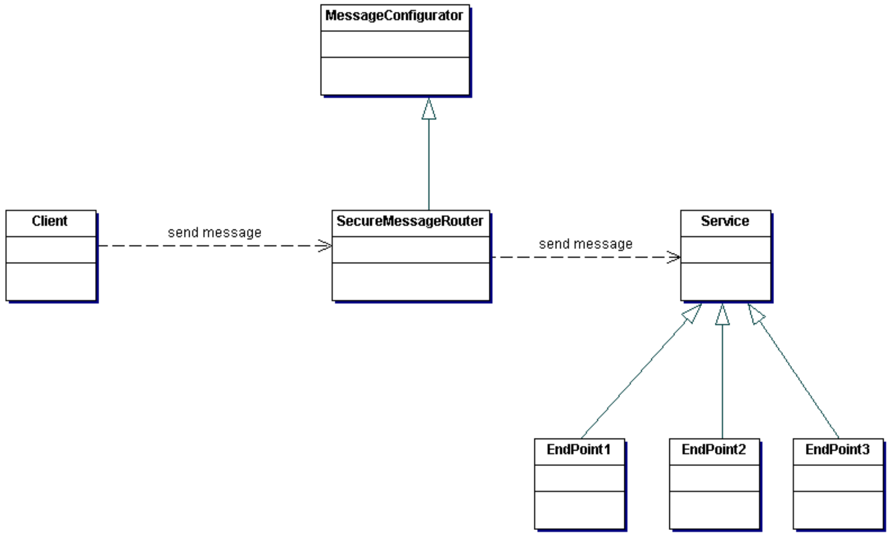
- You need a security intermediary for Web services based workflow or multiple service endpoints.
- You need to configure element-level security and access control
- You want to revealing only the required portions of a protected message to a target recipient.
- You want to enable SSO (Ex: generating 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

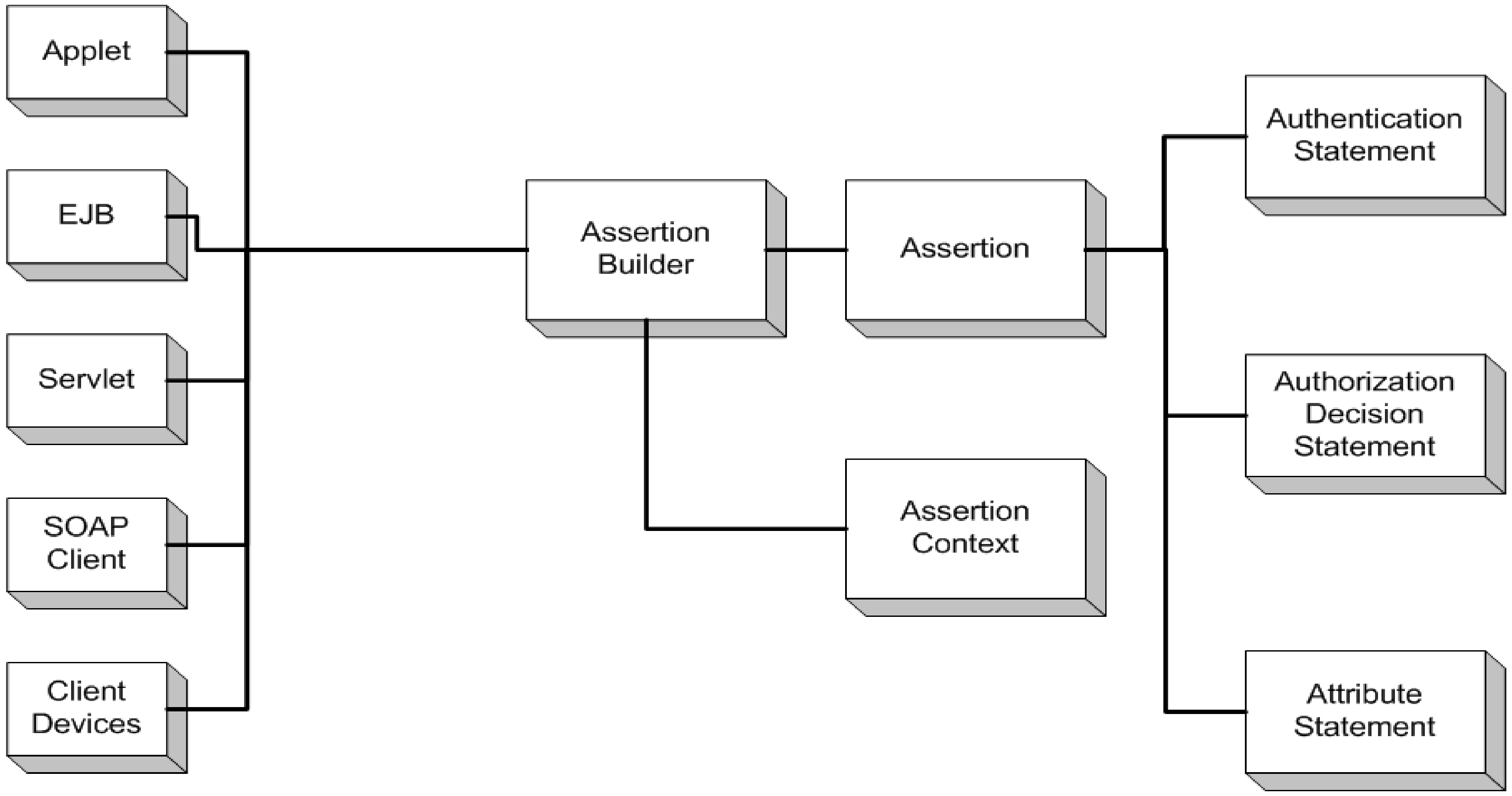
You need a structured and consistent approach to gather security information (for example, SAML assertions) about the authentication action performed on a subject, attribute information about the object, or an authorization request from a trusted service provider.

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

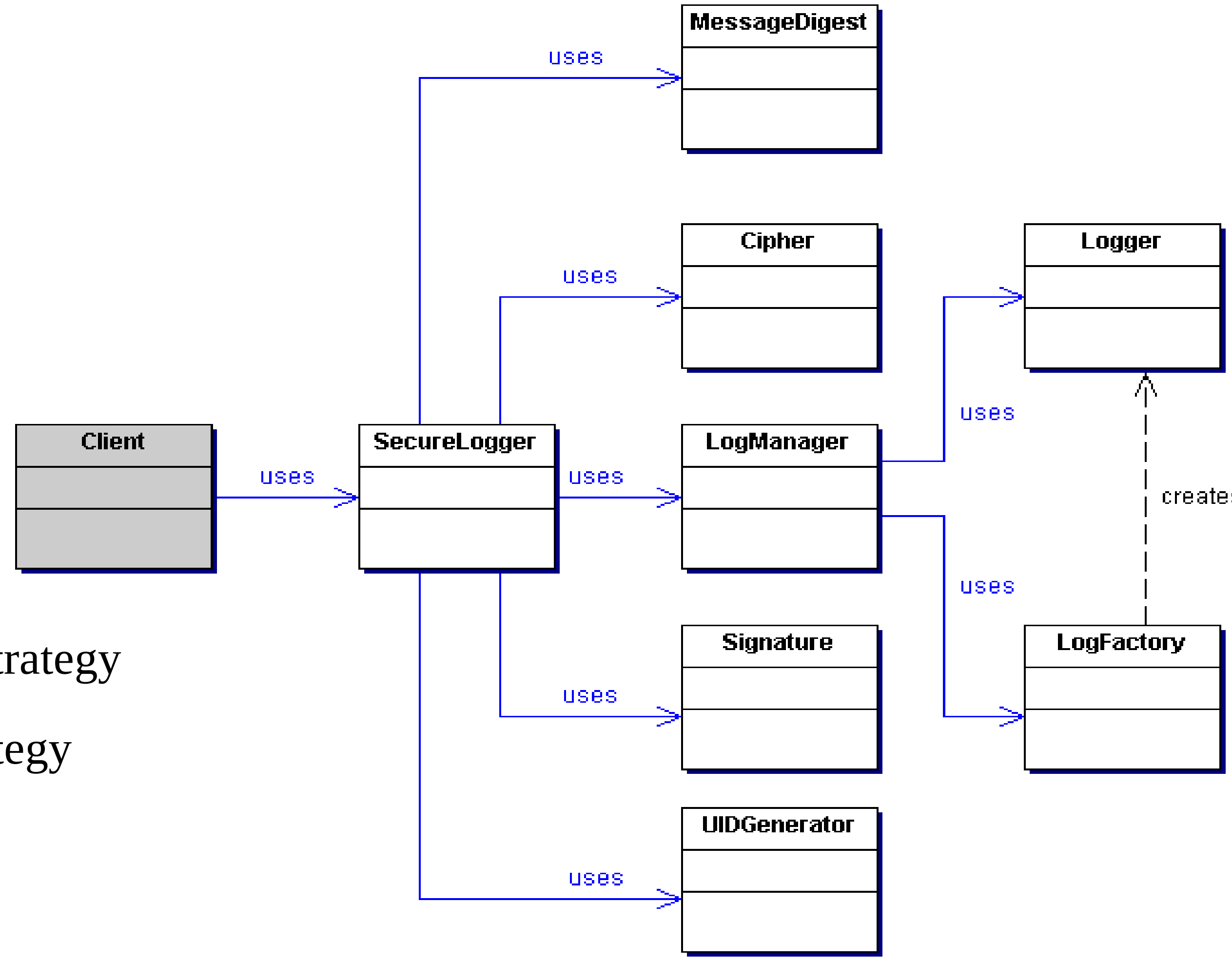
Problem

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

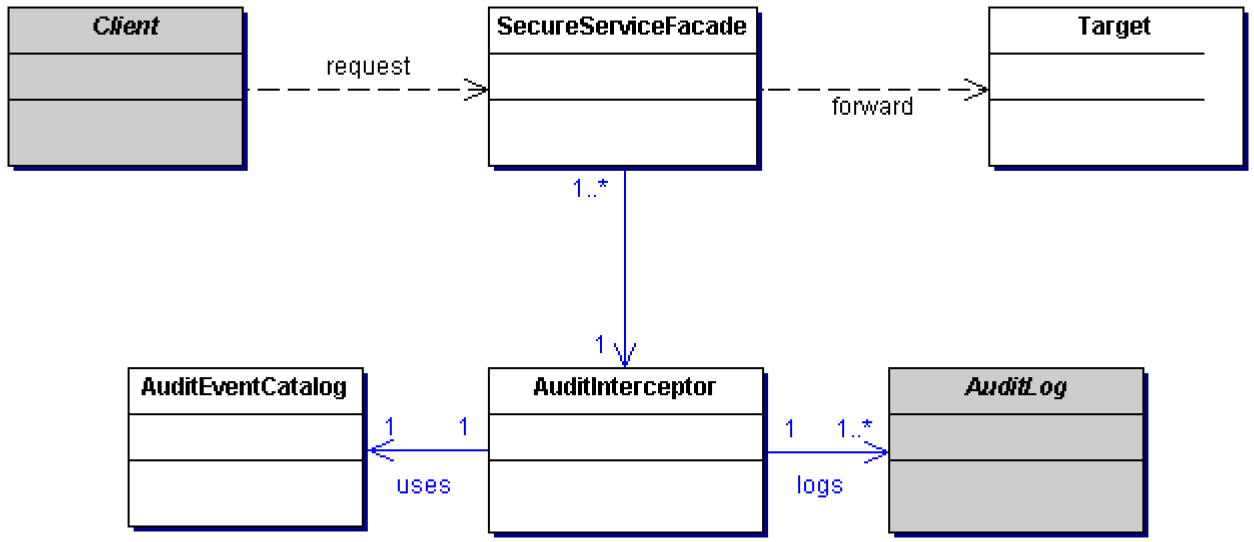
Problem

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

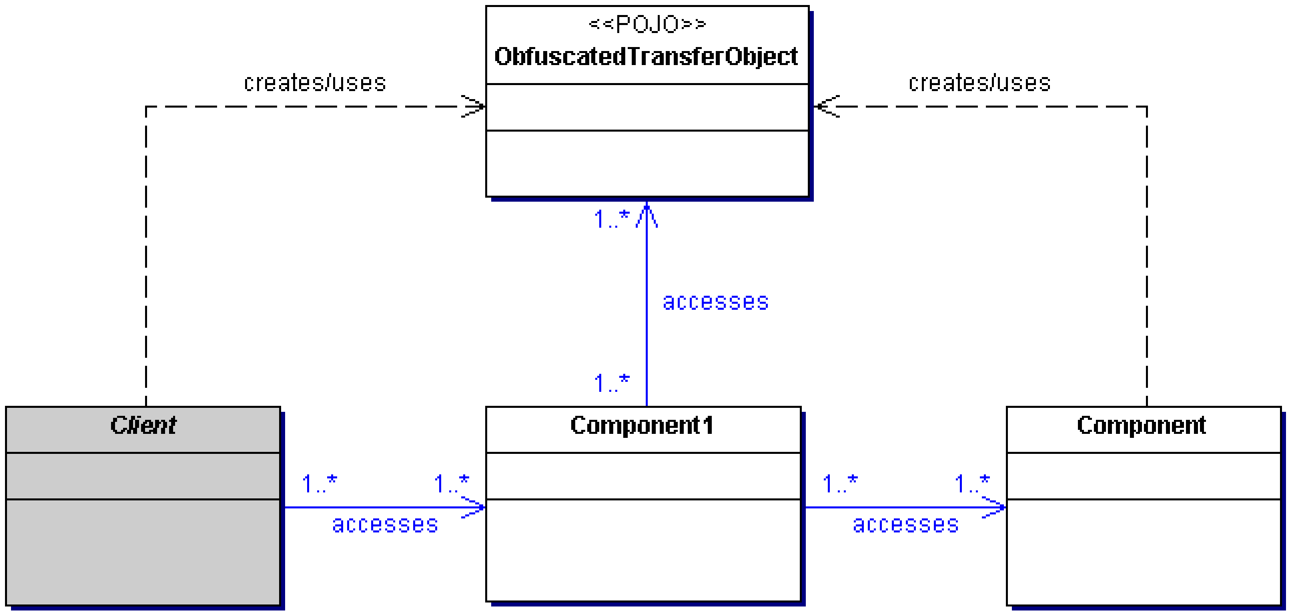
Problem

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

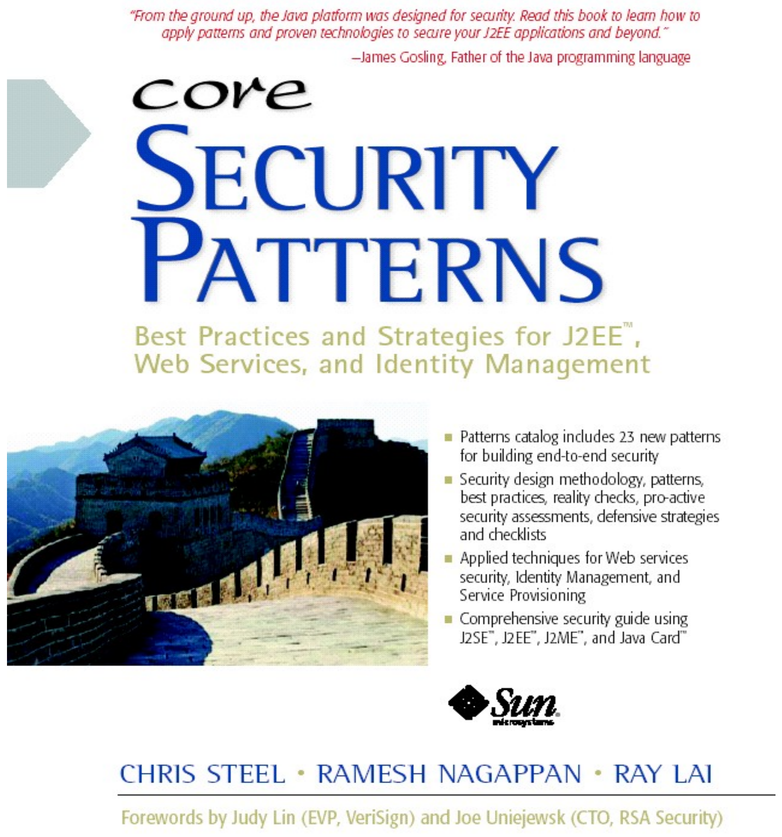
1. Choose and implement standards-based security infrastructure components.
2. Protect your network perimeter with firewalls, router ACLs and IDS.
3. Enforce communication security ensuring confidentiality and integrity.
4. Choose an XML-aware security infrastructure at the perimeter level
5. Restrict 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

Sun microsystems

Further Reading *(Shameless Plug)*

- **Core Security Patterns: Best Practices and Strategies for J2EE, Web Services and Identity Management**
- **Visit CoreSecurityPatterns.Com for more information and resources.**
- **Feedback to authors@coresecuritypatterns.com**



"From the ground up, the Java platform was designed for security. Read this book to learn how to apply patterns and proven technologies to secure your J2EE applications and beyond."
—James Gosling, Father of the Java programming language

core
SECURITY
PATTERNS

Best Practices and Strategies for J2EE™, Web Services, and Identity Management

- Patterns catalog includes 23 new patterns for building end-to-end security
- Security design methodology, patterns, best practices, reality checks, pro-active security assessments, defensive strategies and checklists
- Applied techniques for Web services security, Identity Management, and Service Provisioning
- Comprehensive security guide using J2SE™, J2EE™, J2ME™, and Java Card™

Sun microsystems

CHRIS STEEL • RAMESH NAGAPPAN • RAY LAI

Forewords by Judy Lin (EVP, VeriSign) and Joe Uniejewsk (CTO, RSA Security)

Thank You

Ramesh Nagappan CISSP

Christopher Steel CISSP

authors@coresecuritypatterns.com