



Security Patterns

95-706 Object Oriented
Analysis & Design

Sasha Romanosky

May 01, 2006

Whoami

- Security guy at heart
- 8+ years working in financial, ecommerce industry
- Author of security patterns and other infosec topics
- Most recently at eBay doing vulnerability management and phishing investigations
- Current MSISPM student, future PhD student (looking at the economics of information security)

About Patterns

- Developed in the 1970's by Christopher Alexander He observed relationships that existed between things: objects, spaces, light, people, passages, moods
- Later adapted to OO programming, information security
- They are **proven** solutions to **reoccurring** problems
- They work together to form pattern languages
- A couple of important points about patterns (especially if you ever consider writing some):
 - Very hard to write well. A “great idea” is not a pattern
 - Patterns don't represent “new” work (the way most papers do)
 - They codify existing knowledge to help non-experts implement good solutions

About Patterns (2)

They are as much about format as content – perfect for those who love to have things organized.

The 3 part rule:

- **Context:** describes the general conditions under which the problem occurs
- **Problem:** describes the problem that repeatedly occurs and the forces that exist within the context. Forces may complement or contradict one another.
- **Solution:** shows how to best solve the reoccurring problem, or better, how to balance the forces associated with the problem

Mini-Pattern Language

- Single Access Point
- Check Point
- Role Based Access Control (RBAC)

Single Access Point

(Yoder and Baraclow, 1998, 2006)

Context: You have a system that needs to be accessed by users and you want to be able to control how they first interact with it.

Problem: How can you design a system to provide efficient and secure access?

Forces

- Your application may be made up of many parts, and you need to provide a singular entry point for all of them
- Many entry points reduce security because of additional complexity and duplicate efforts necessary to protect each entry point
- The application may require more sophisticated authentication and authorization facilities

Single Access Point (2)

Solution

- Provide a single point where users access the system, and protect the rest of the system's boundaries
- Make this prominent to users
- Enforce business or security policies with the Check Point Pattern

Consequences

- Provides a clearly defined entrance for users of the system
- You can now monitor who comes and goes (for logging, reporting, auditing, marketing)
- Possible privacy risks. Fine for you as an organization, but how will the users feel?
- Can be a single point of failure

Check Point

(Yoder and Baraclow, 1998, aka PEP, PDP, SSO)

Context: You have implemented Single Access Point, and now you need to enforce security policies on your users

Problem: How can you design a system to efficiently grant access to legitimate users?

Forces

- Users need to be authenticated and authorized in order to access the resources for which they are entitled
- Authorization is generally application specific, but authentication is not
- Implementing this at every step is cumbersome and inefficient, and makes it difficult to modify

Check Point (2)

Solution

- Create an object that encapsulates the organization's security policy – the Session pattern
- Separate authentication and authorization processes
- Abstract authentication from the application (like MVC)
- Grant access to resources based on the user's role – RBAC

Consequences

- General access can now be defined and modified in a single place
- Can now authenticate against many disparate user stores (think SSO)
- Some applications may still require multiple Check Points
- Authentication from disparate systems (website, pda, cellphone) enter each through their own Single Access Point, but all communicate with the Check Point

Role Based Access Control

(Yoder and Baraclow, 1998, Fernandez 2006)

Context: You are designing a computer application. You have implemented the Single Access Point and Check Point patterns and now you need to provision access to resources for many users.

Problem: How do you provision access to many users based on their job function?

Forces:

- Often, people can be classified according to their job function
- There may be too many access rules to keep track of individually. ACLs are no longer practical.
- Some people share many responsibilities, but others do not

Role Based Access Control (2)

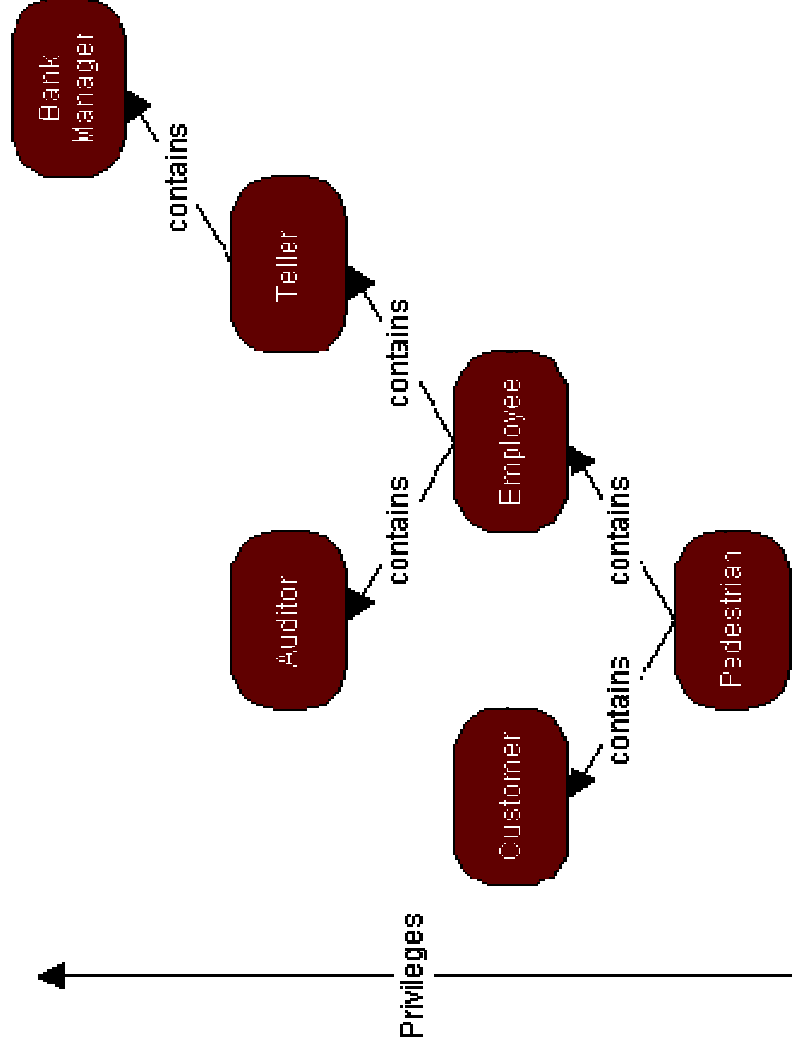
Solution:

- Abstract the user from their privileges
- Assign users to roles
- Assign unique collection of privileges to those roles

Consequences:

- Roles are designed around business functions, making them more intuitive to manage
- Concept of least privileges can be enforced because no role is granted more privileges than allowed
- Now you can define policies to implement separation of duties (static, dynamic, organizational)
- Inheritance is easy to implement

RBAC Example: The Bank

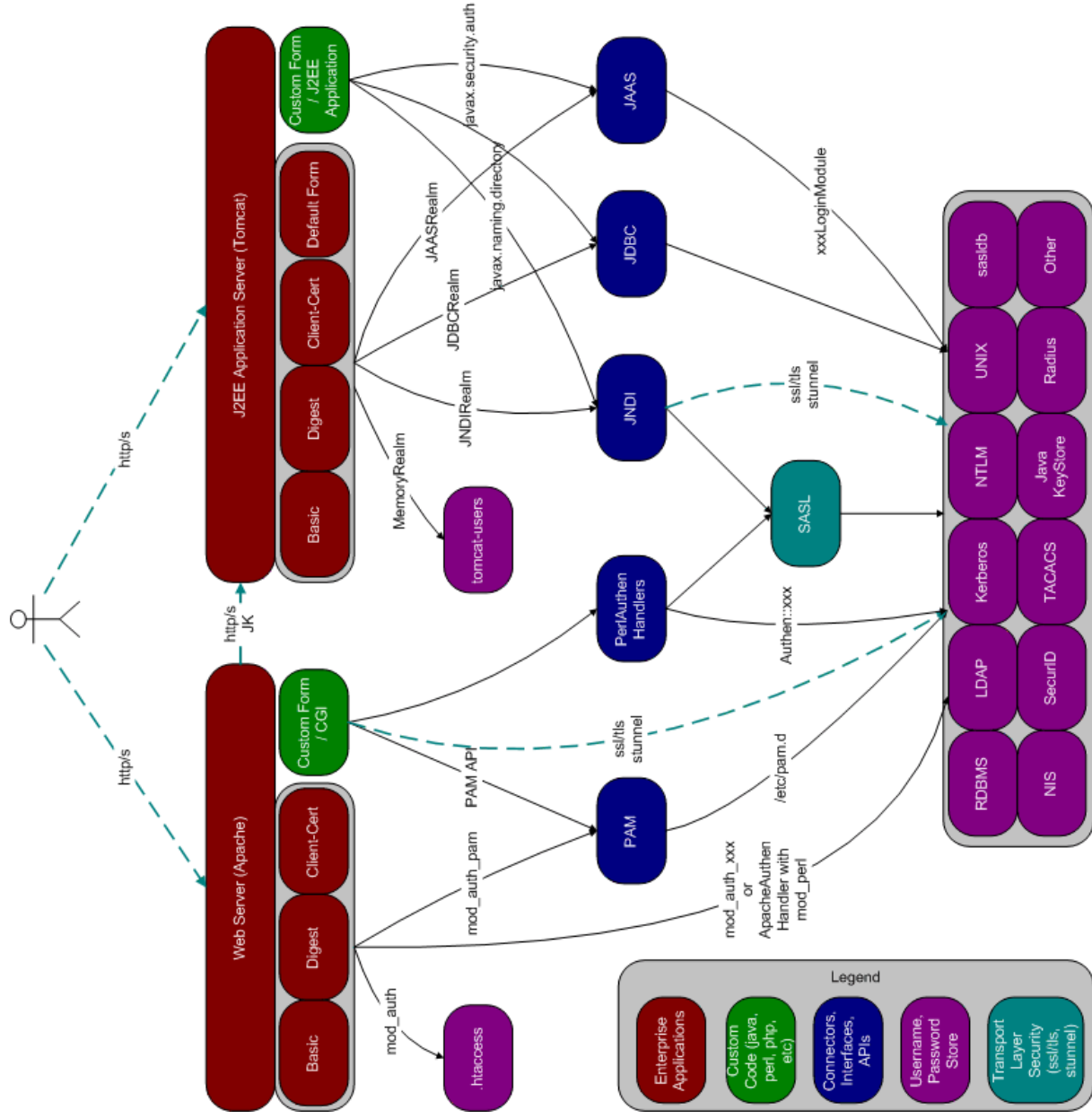


Role Based Access Control (3)

Example: A bank has customers, tellers, an Auditor and a manager

- **Least Privileges:** no user is granted more access than necessary
- **Inheritance:** Roles are define in a hierarchy. The role “above” inherits all the privileges of the role “below”
- **Static separation of duties:** No user can belong to two particular roles, ever. E.g. a user could never be both a bank manager and an auditor
- **Dynamic Separation of Duties:** No user can belong to two particular roles during the same session. E.g. a user could never be both a customer and a teller *at the same time*

Authentication Options for a Web-based Application



Where to learn more

- **www.hillside.net**. They host a tour of international conferences on patterns: the Pattern Language of Programs (PLoPs).
- **www.securitypatterns.org**: books, blog, other references

