中山大學
SUN YAT-SEN UNIVERSITY

Module III. Web Security

Chapter 8
Web Security Primer

Web Security: Theory & Applications

School of Data & Computer Science, Sun Yat-sen University

Outline

- **8.1 Overview**
  - C/S and B/S Models
  - Web Site Architecture
  - Electronic Commerce Architecture
  - Apache & IIS
- **8.2 Web Services**
  - SOA
  - Web Services
  - SOAP
- **8.3 Web Security Primer**
  - Web Security - Beginning
  - Web Server Vulnerabilities
  - Web Services Secure Model
  - Prevention of Malicious Codes
  - SSL/HTTPS

8.1 Overview

8.1.1 C/S and B/S Models

❑ Definition of Web Applications

- A web application is an application that is accessed over a network such as the Internet or an intranet.
- Wikipedia
  - ✧ In computing, a web application or web app is a client–server software application in which the client (or user interface) runs in a web browser.
  - ✧ Web Application Security is a branch of Information Security that deals specifically with security of websites, web applications and web services.

8.1 Overview

8.1.1 C/S and B/S Models

❑ Definition of Web Applications

- Web 的组成部分
  - ✧ 服务器端 (Web 服务器)
    - ◌ 在服务器结构中规定了服务器的传输设定、信息传输格式以及服务器本身的基本开放结构。
  - ✧ 客户端 (Web 浏览器)
    - ◌ 客户端通常称为 Web 浏览器，用于向服务器发送资源请求，并将接收到的信息解码显示。
  - ✧ 通讯协议 (HTTP协议)
    - ◌ HTTP (HyperText Transfer Protocol，超文本传输协议) 是分布式的 Web 应用的核心技术协议。它定义了 Web 浏览器向 Web 服务器发送索取 Web 页面请求的格式，以及 Web 页面在互联网上的传输方式。

8.1 Overview

8.1.1 C/S and B/S Models

❑ Definition of Web Applications

- Web 的组成部分
  - ✧ Web 应用程序
    - ◌ 即基于 Web 的应用程序。
    - ◌ 由 web 站点及其它资源组成，包括数据库等各种数据文件。典型的 web 应用程序基于 B/S 模式开发，以 web 服务器为支持，通过与浏览器的通讯，响应用户操作。

8.1 Overview

8.1.1 C/S and B/S Models

❑ History of Web Applications

- Early days
  - ✧ 每个网页作为一个静态文件被传递到客户端，网页变化需要往返回服务器刷新整个页面。
- In 1995
  - ✧ Netscape 引入了 JavaScript，允许程序员添加一些动态元素到客户端上运行的用户界面。
- In 1996
  - ✧ Macromedia 引入 Flash，它允许使用脚本语言在客户端交互，无需与服务器通信。

8.1 Overview

8.1.1 C/S and B/S Models

❑ History of Web Applications

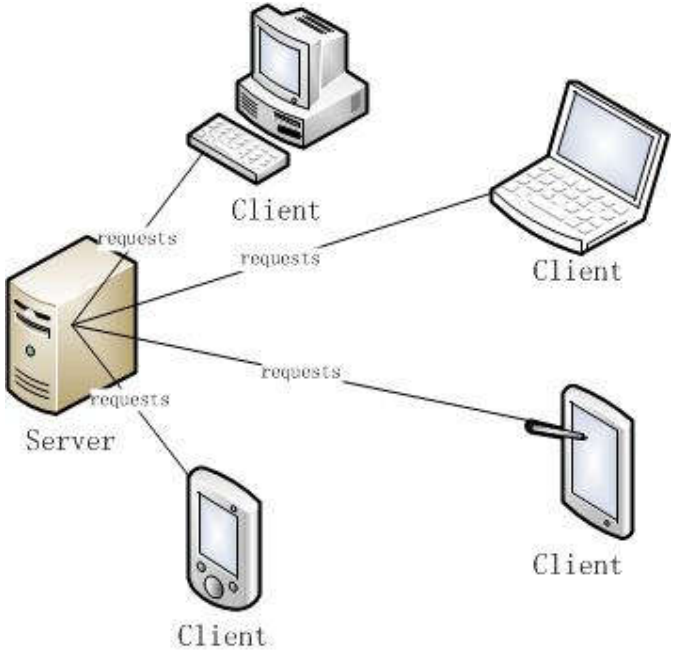
- In 1999
  - ✧ Java Servlet 2.2 becomes part of J2EE, the concept of independent web applications introduced.
- In 2005
  - ✧ Ajax 一词被创造出来，可以在不重新加载整个网页的情况下，对网页的某部分进行更新。
- In 2011
  - ✧ HTML5 完成定稿，它提供了图形和多媒体功能且无需客户端插件。

8.1 Overview

8.1.1 C/S and B/S Models

❑ C/S Model

- What is C/S Model
  - ✧ The **client–server model** of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.
  - ✧ Both client and server are processes.
  - ✧ Both client and server may reside in the same system.



8.1 Overview

8.1.1 C/S and B/S Models

❑ C/S Model

- 客户机-服务器结构模型
  - ✧ C/S 结构是一种软件系统体系结构，目的在于充分利用两端硬件环境的优势，将任务合理分配到 Client 端和 Server 端实现，降低系统的通讯开销。
  - ✧ C/S 结构将计算机应用任务分解成多个子任务，由多台计算机分工完成，实现“功能分布”原则。 Client 端完成数据表示、数据处理以及用户接口功能； Server 端完成 DBMS (数据库管理系统) 的核心功能。
  - ✧ Client 端和 Server 端可以在地理位置上分离，也可以处于同一台主机内部，而只是功能逻辑上分离。

8.1 Overview

8.1.1 C/S and B/S Models

❑ C/S Model

- 客户机-服务器结构模型
  - ✧ 传统的 C/S 体系结构虽然采用的是开放模式，但只是系统开发一级的开放性。在特定的应用中，无论是 Client 端还是 Server 端都需要特定的软件支持。由于未提供用户真正期望的开放环境，C/S 结构需要针对不同的操作系统系统开发不同版本的软件，加之产品快速的更新换代，要达到对需求的及时适应需要耗费很高的成本。

8.1 Overview

8.1.1 C/S and B/S Models

❑ C/S Model

- C/S 模式的优点
  - ✧ 客户端与服务器直接相连，没有中间环节，因此可以提供高的响应速度。
  - ✧ 操作界面按需设计、形式多样，可以充分满足客户自身的个性化要求。
  - ✧ C/S 结构的管理信息系统具有较强的事务处理能力，能够实现复杂的业务流程。
  - ✧ C/S 结构可以对用户权限进行多层次校验，提供了更高的安全性能保证。

8.1 Overview

8.1.1 C/S and B/S Models

❑ C/S Model

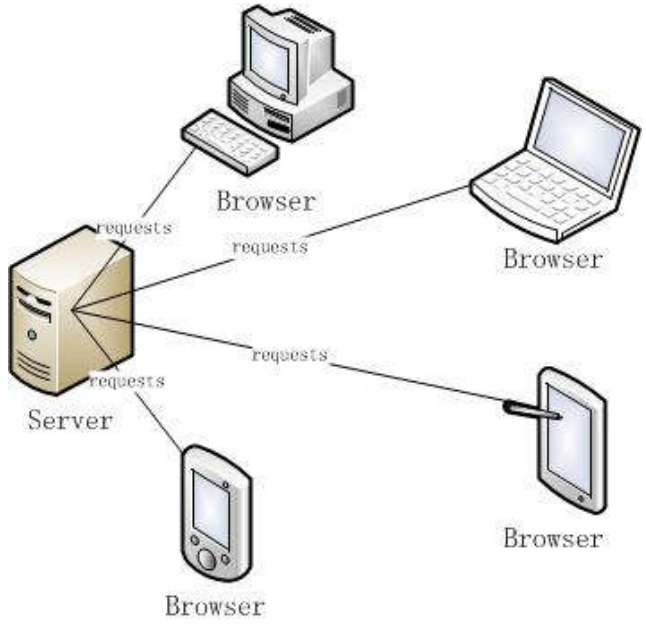
- C/S 模式的缺点
  - ✧ 需要专门的客户端程序，分布功能弱。针对点多面广且不具备网络条件的用户群体，不能够实现快速部署安装和配置。
  - ✧ 兼容性差。系统的实现与开发工具密切相关，具有较大的局限性。开发工具发生变革时，需要重新改写程序。
  - ✧ 开发成本高。需求分析和按需设计耗费大量的开发周期时间，对开发人员的专业水准也有较高的要求。

8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- What is B/S Model
  - ✧ The **browser–server model** of computing is an improvement of C/S model. With this architecture, browsers realize the user interface and servers deal with the main business logic.



8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- 浏览器-服务器结构模型
  - ✧ B/S 结构是随着因特网技术的兴起，对 C/S 结构的一种变化或者改进的结构。在 B/S 结构下，用户工作界面通过浏览器实现，只有极少部分的事务逻辑在前端 (Browser) 实现，主要事务逻辑在服务器端 (Server) 实现，形成所谓三层应用结构 (3-tier application, 即表现层 UI、业务逻辑层 BLL、数据访问层 DAL)，大大减少了客户端的电脑载荷，减轻了系统维护与升级的工作量，降低了用户的总体成本 (TCO, Total Cost of Ownership)。
  - ✧ B/S 结构是一次性到位的开发，此外实现了不同的人员，从不同的地点，以不同的接入方式 (比如 LAN, WAN, Internet /Intranet 等) 访问和操作共同的数据库。在权限管理和服务器数据库安全保护方面也更为有效。

8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- B/S 结构的优点
  - ✧ 具有分布性特点，可以随时随地进行查询、浏览等业务处理。
  - ✧ 业务扩展简单。通过增加服务器端网页即可增加服务器功能。
  - ✧ 维护方便。只需要改变服务器端网页，即可实现所有用户功能的同步更新。
  - ✧ 开发周期短，共享性强。

8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- B/S 模式的缺点
  - ✧ 个性化特点降低，无法实现具有个性化的功能要求。
  - ✧ 操作在浏览器界面上进行，无法满足快速操作的要求。
  - ✧ 页面动态刷新，响应速度明显降低。
  - ✧ 功能弱化，难以实现传统模式下的特殊功能要求。

8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- **C/S vs. B/S**

C/S	B/S
Advantage	**Advantage**
1. High speed responsibility 2. Individual UI 3. Ability to deal with business logic	1. Less limitation of time and space 2. Fine augmentability (可扩展性) 3. Simple maintenance 4. Easy/quick development
Disadvantage	**Disadvantage**
1. Client software 2. Bad compatibility (兼容性) 3. High developing cost	1. Changeless UI 2. Low speed responsibility 3. Poor function

8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

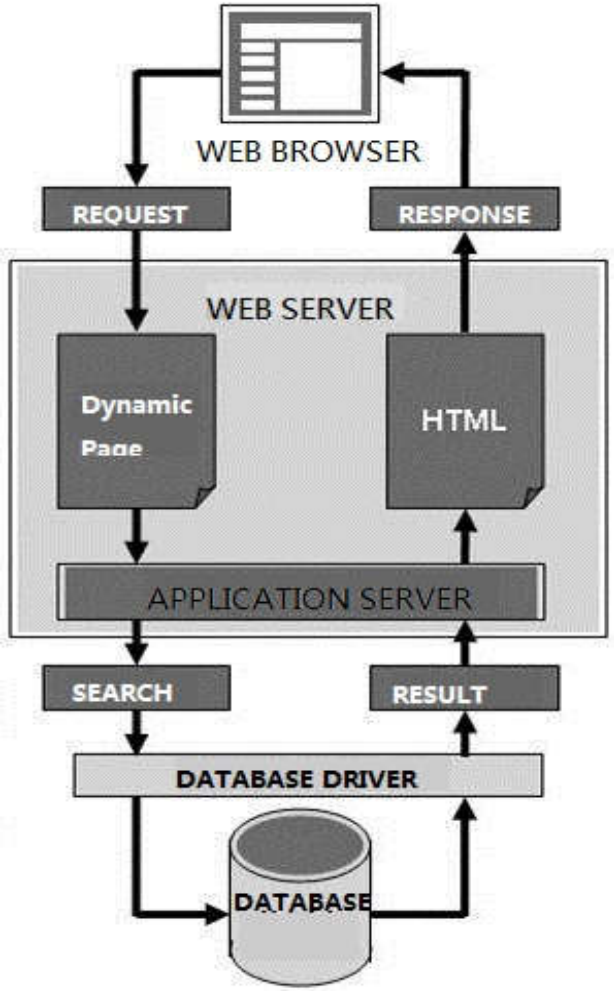
- C/S 结构和 B/S 结构的比较
  - ✧ C/S 技术源于上世纪90年代初，其结构经历了二层 C/S、三层 C/S 的更迭，技术应用已经相当成熟。B/S 技术则是伴随着因特网的普及而来的，通常称为 Web 应用。
  - ✧ B/S 和 C/S 都是当前非常重要的计算架构。在适用因特网体系、降低维护成本等方面，B/S 比 C/S 要强；但在运行速度、数据安全、人机交互等方面，C/S 则有明显的优势。例如，对于以浏览为主、录入简单的应用程序，B/S 技术有很大的优势；而对于交互复杂的 ERP 等企业级应用，目前未见成熟的 B/S 架构。从全球范围看，成熟的 ERP 产品大多采用二层或三层 C/S 架构，且有向 SOA 发展的趋势。

8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- Basic structure for Web applications



8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- Web 应用的基本结构
  - ✧ Web 应用程序通过 Web 浏览器与 Web 服务器进行通讯。Web 服务器对用户透明，用户仅通过与浏览器进行交互，便可获得所需信息，完成对 Web 应用程序的所有功能性需求。对服务器而言，用户的操作仅仅是浏览器返回的一组数据，服务器接收并处理这组数据，然后向浏览器返回相应的信息。因此，一个最基本的 Web 应用，需要包含以下四个过程：浏览器接收用户操作；浏览器向服务器起请求；服务器响应该请求；浏览器向用户反馈信息。
  - ✧ 根据问题的复杂程度，Web 应用结构可以分为：静态网页、动态网页、带数据库的动态网页。

8.1 Overview

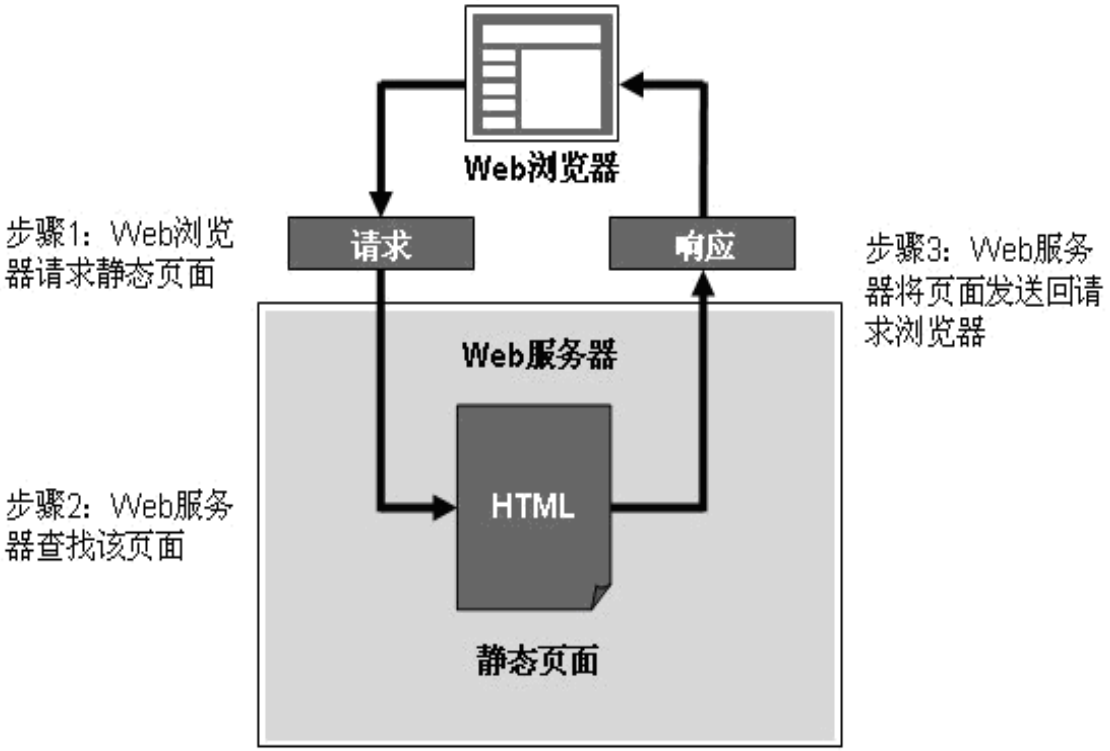
8.1.1 C/S and B/S Models

- **B/S Model**
  - 静态网页
    - 静态网页是事先用 HTML 编写好、内容固定不变的文件，通过上载等方式保存在 Web 站点的可访问文件夹中。当用户浏览器向 Web 服务器发出访问该页面的请求时，服务器在站点内查找该页面并直接将其下载给客户浏览器。静态网页无论在 Web 站点还是下载到用户的浏览器上，其内容都保持一致。
      - 静态网页文件名后缀如 .htm、.html、.shtml

8.1 Overview

8.1.1 C/S and B/S Models

- **B/S Model**
  - 静态网页
    - ✧ 静态网页的访问流程



8.1 Overview

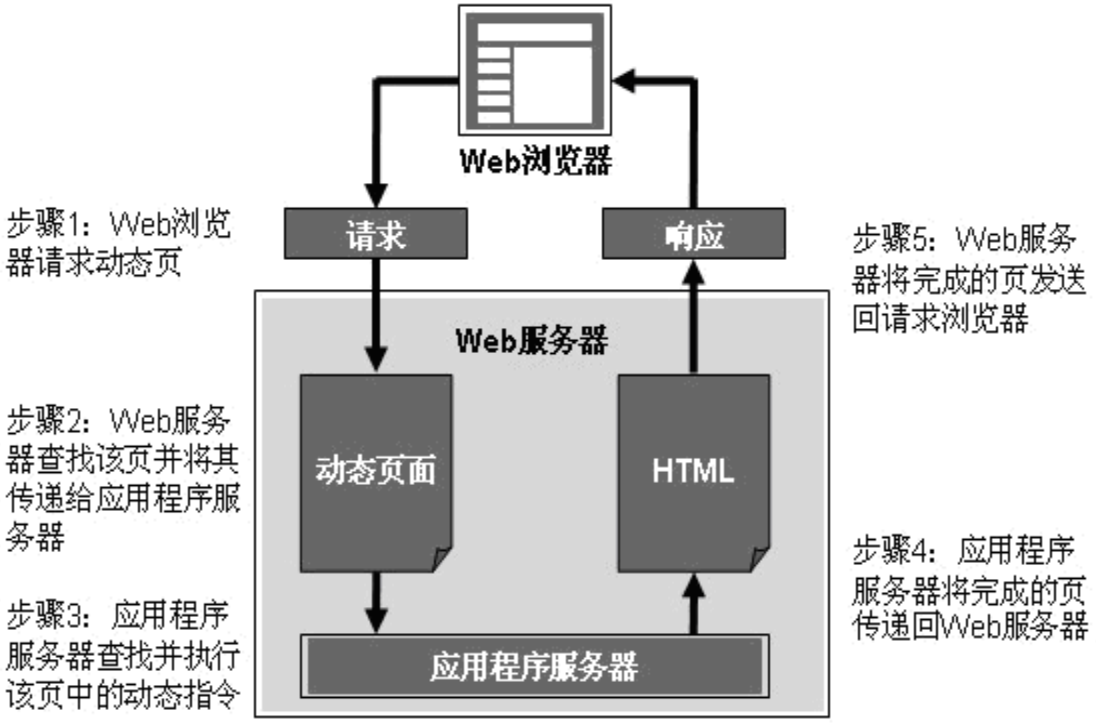
8.1.1 C/S and B/S Models

- **B/S Model**
  - 动态网页
    - 动态网页是由 Web 服务器动态生成的文件。称其为“动态”，是因为网页的部分甚至全部内容都不是事先确定的。用户通过浏览器提出访问该页面的请求时，Web 服务器按照用户请求，通过应用程序服务器对相关服务器端的数据进行处理，根据处理结果生成标准的 HTML 文件下传给用户浏览器。不同的用户请求以及访问数据的变化都会随之生成不同的 HTML 文件下传给用户。
      - 动态网页文件名后缀如 .asp、.jsp、.php、.aspx

8.1 Overview

8.1.1 C/S and B/S Models

- **B/S Model**
  - 动态网页
    - ✧ 动态网页的访问流程



8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

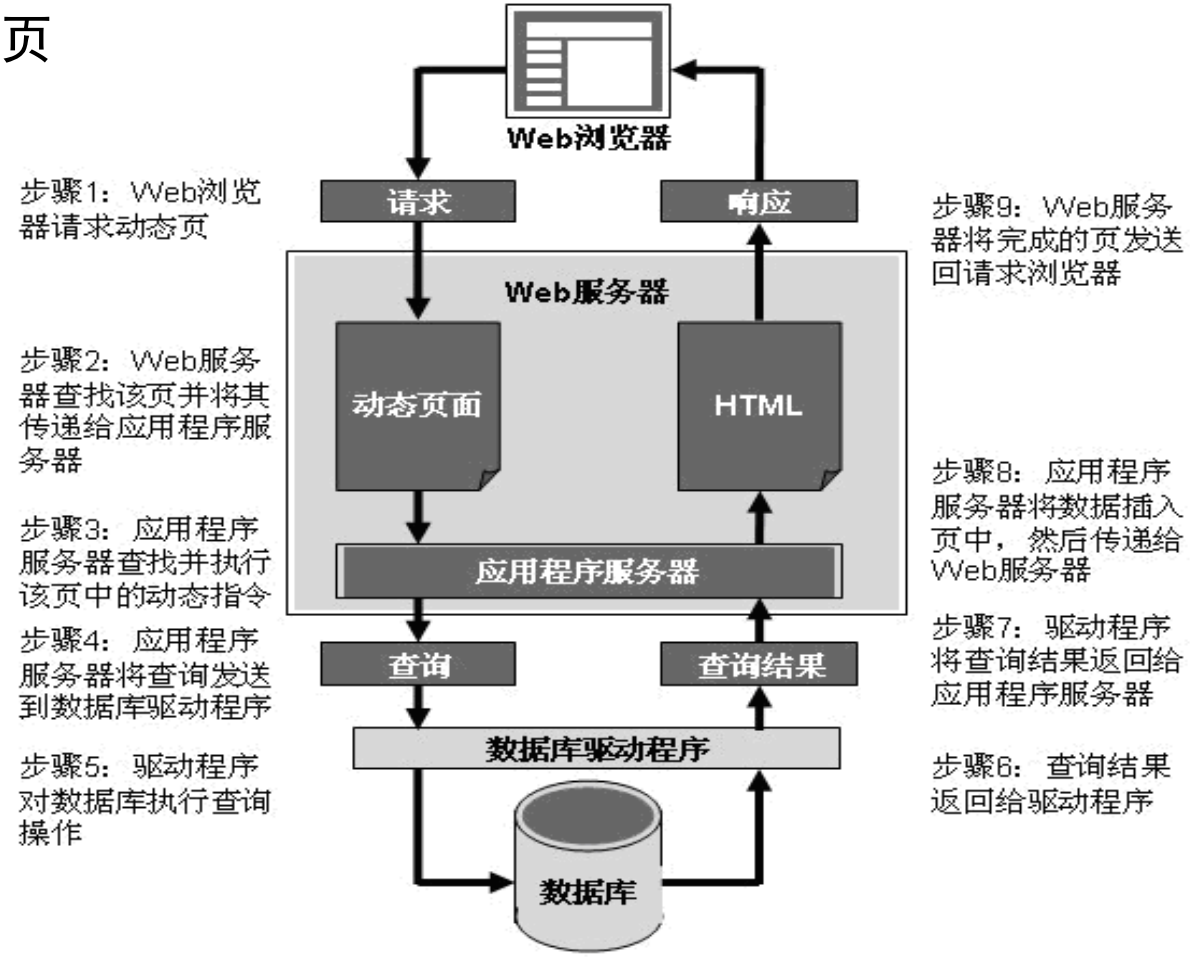
- 带数据库的动态网页
  - ✧ 目前，大多数的动态网页需要存储并处理大量的数据，Web 应用程序的一个重要用途也是用来实现基于数据库的信息管理系统。实现带数据库的动态页面在逻辑上比普通的动态页面多一层，即数据库层。

8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- 带数据库的动态网页
  - ✧ 带数据库的动态页面的访问流程

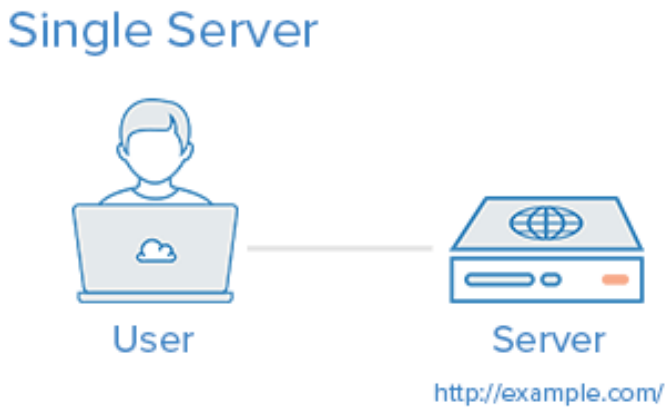


8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- Web Server Setups
  - ✧ Everything on One Server

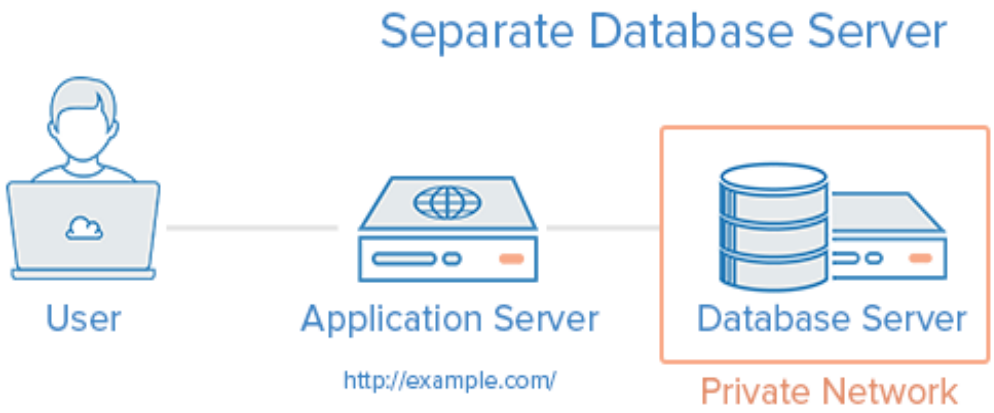


8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- Web Server Setups
  - ✧ Separate Database Server

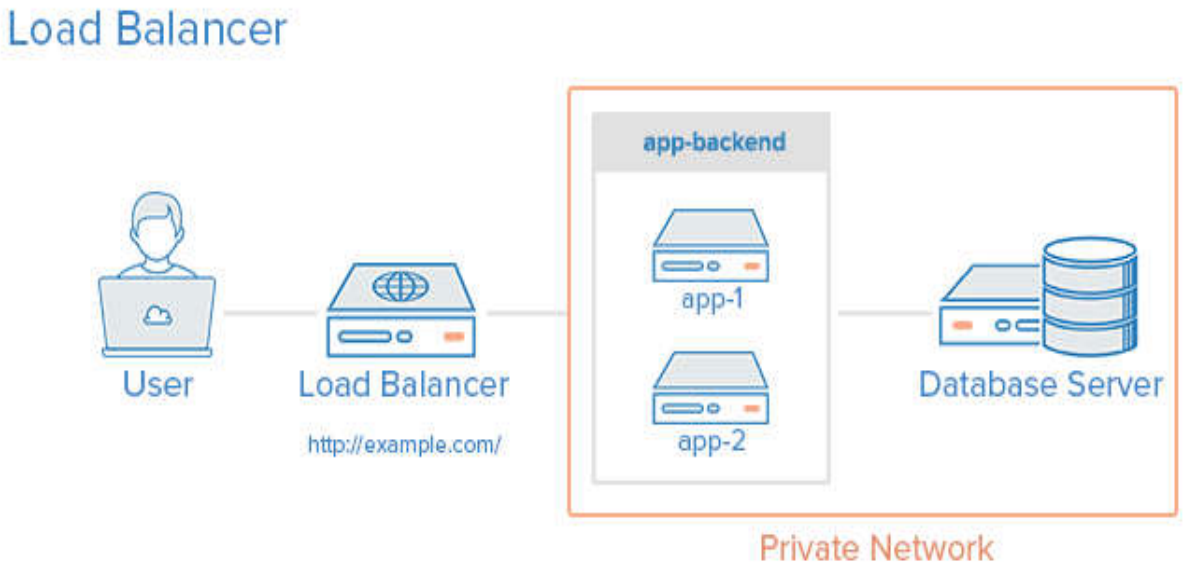


8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- Web Server Setups
  - ✧ Load Balancer (Reverse Proxy)

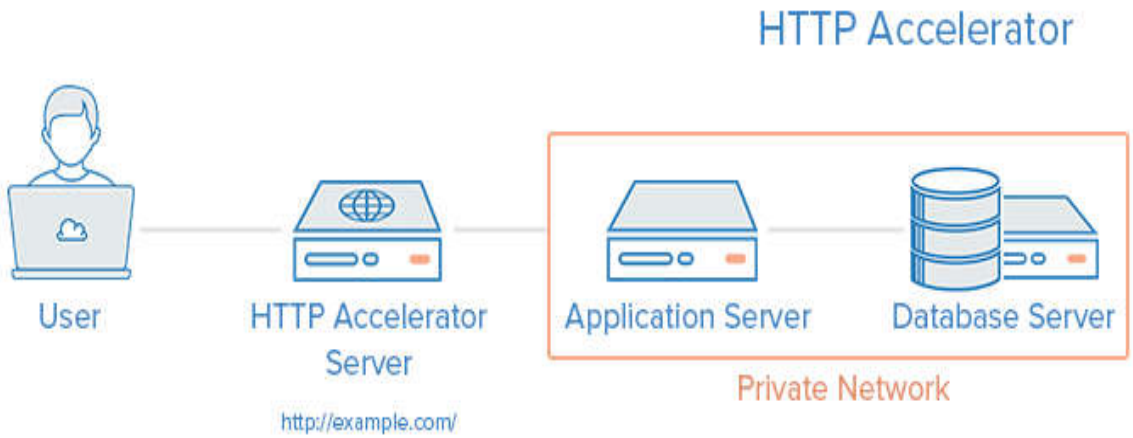


8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- Web Server Setups
  - ✧ HTTP Accelerator (Caching Reverse Proxy)

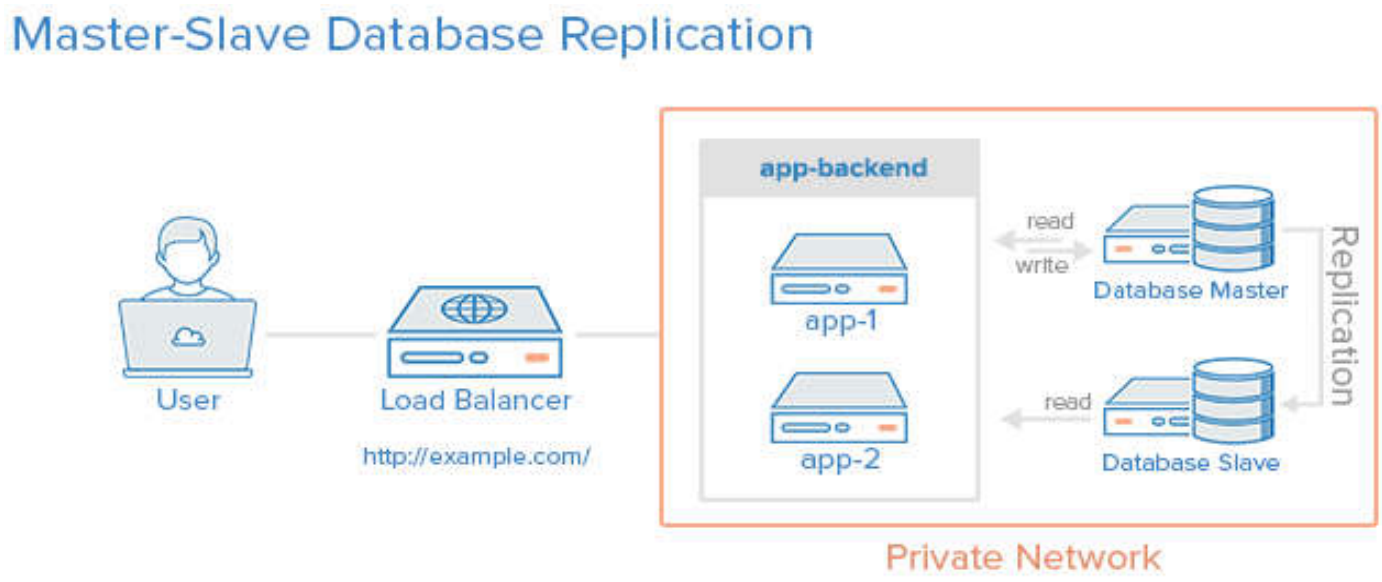


8.1 Overview

8.1.1 C/S and B/S Models

❑ B/S Model

- Web Server Setups
  - ✧ Master-Slave Database Replication

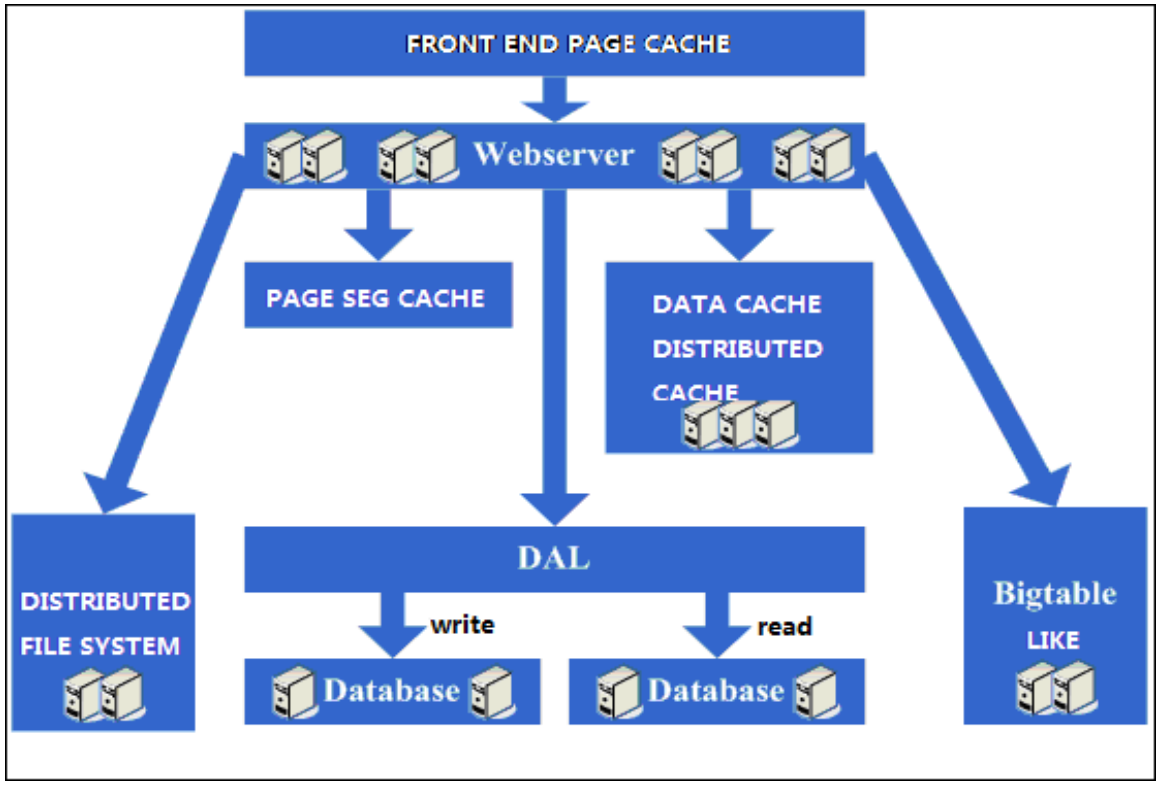


8.1 Overview

8.1.2 Web Site Architecture

❑ Hardware Architecture

- image server, page server, database server, application server, log server…



8.1 Overview

8.1.2 Web Site Architecture

❑ Hardware Architecture

- 硬件架构
  - ✧ 这里的硬件架构单指服务器的组合。对于大型、复杂、性能要求高的网站，需要为每一种特定的服务指定一台专用的服务器或一个专用的服务器集群。服务器可能包括：图片服务器，页面服务器，数据库服务器，应用服务器，日志服务器等等。
    - ◌ 数据库服务器是重点设备，需要慎重选择和配置。
    - ◌ 页面服务器只保存静态内容，PHP 应用服务器设置用于执行脚本。
    - ◌ 大访问量 (百万级) 的网站需要设立独立的日志服务器。
    - ◌ 访问量大的网站可以采用分离的图片服务器和页面服务器，例如用 lighttpd 作为图片服务器，用 apache 作为页面服务器。

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- The three-tier architecture model (*John J. Donovan*) is a C/S software architecture pattern. Being the fundamental framework for the logical design model, it segments an application's components into three tiers of services in which the user interface (Presentation Tire), functional process logic (Business Logic Tier), computer data storage and data access (Data Tier) are developed and maintained as independent modules, most often on separate platforms.
- These tiers do not necessarily correspond to physical locations on various computers on a network, but rather to logical layers of the application. How the pieces of an application are distributed in a physical topology can change, depending on the system requirements.
- During an application's life cycle, the three-tier approach provides benefits such as reusability, flexibility, manageability, maintainability, and scalability.

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- Presentation Tier
  - ✧ The presentation tier, or user services layer, gives a user access to the application. This layer presents data to the user and optionally permits data manipulation and data entry.
  - ✧ The two main types of user interface for this layer are the traditional application and the Web-based application.
  - ✧ Web-based applications now often contain most of the data manipulation features that traditional applications use. This is accomplished through use of Dynamic HTML and client-side data sources and data cursors.

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- Business Logic Tier
  - ✧ The middle tier, or business services layer/application layer, also referred to as the *business logic tier*, consists of business and data rules. This tier is where COM+ developers can solve mission-critical business problems and achieve major productivity advantages. The components that make up this layer can exist on a server machine, to assist in resource sharing. These components can be used to enforce business rules, such as business algorithms and legal or governmental regulations, and data rules, which are designed to keep the data structures consistent within either specific or multiple databases. Because these middle-tier components are not tied to a specific client, they can be used by all applications and can be moved to different locations, as response time and other rules require.

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- Data Tier
  - ✧ The data tier, or data services layer/persistent layer, interacts with persistent data usually stored in a database or in permanent storage. This is the actual DBMS access layer. It can be accessed through the business services layer and on occasion by the user services layer. This layer consists of data access components (rather than raw DBMS connections) to aid in resource sharing and to allow clients to be configured without installing the DBMS libraries and ODBC (open database connectivity) drivers on each client.

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- Web service built using three tiers
  - ✧ A front-end web server serving static content, and potentially some cached dynamic content. In web based application, front-end is the content rendered by the browser. The content may be static or generated dynamically.
  - ✧ A middle dynamic content processing and generation level application server, for example, Java EE, ASP .NET, PHP, Perl platform.
  - ✧ A back-end database or data store, comprising both data sets and the DBMS or RDBMS software that manages and provides access to the data. (Wikipedia)

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- Web service built using three tiers
  - ✧ 3-Tier Web Architecture is that unique system of developing web database application which works around the 3-tier model, comprising of database tier at the bottom, the application tier in the middle and the client tier at the top. This comprehensive 3-tier architecture module is the framework for most Web Applications on the Internet. This system helps to separate the Business Logic from the Application, Data Storage and database.
  - ✧ 3-tier Web Architecture is designed to provide a greater degree of flexibility and increased security that can be designed for each service at each level. This unique system of framework for web application development with the 3-tier web architecture also ensures that there is increased performance as the task is shared between servers. 3-Tier Web Architecture is a connection and composition of the three links that facilitates the smooth functioning of the website.

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- Web 服务软件的三层逻辑架构
  - ✧ 软件系统发展到一定规模，需要对软件逻辑进行分层以便维护和扩展。通常 Web 服务的软件架构具备三个层面 (tiers)，自上而下分为表现层 (Presentation Layer)、业务逻辑层 (Business Layer, 包括应用层和领域层) 以及持久层 (Persistent Layer)。
  - ✧ 表现层 (Presentation Layer/UI)
    - ◌ 所有和表现相关的逻辑，包括表现层模板，都属于表现层的范畴。需要避免在非表现层代码里完成本属于表现层逻辑的设计内容。
    - ◌ 表现层负责向用户展现信息以及解释用户命令：
      - 向应用层发出请求以获取用户所需要展现的数据；
      - 向应用层发送命令要求其执行某个用户命令。

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- Web 服务软件的三层逻辑架构
  - ✧ 业务逻辑层 (Business Logic Layer/BLL)
    - ◌ 应用层：定义用户可以“做什么”，并把操作结果反馈给表现层，而将“如何做”通过委派交给领域层处理。
      - 定义软件要完成的所有任务。对外为表现层提供各种应用功能 (包括查询或命令)，对内调用领域层 (领域对象或领域服务) 完成各种业务逻辑。
      - 应用层不包含业务逻辑。
      - 在 MVC (Model-View-Controller) 架构中，应用层相当于 Controller，应该尽量简单，不要包括涉及领域内容的逻辑。

8.1 Overview

8.1.2 Web Site Architecture

- **Software: in 3-tier Architecture**
  - Web 服务软件的三层逻辑架构
    - 业务逻辑层 (Business Logic Layer/BLL)
      - 领域层：包含领域 (Domain) 逻辑的层。应用层和领域层通常在硬件上并不分离。
        - 领域层表达业务概念、业务状态信息以及业务规则，领域模型处于领域层，是业务软件的核心。
        - 引入领域服务可以有效防止领域层逻辑泄露到应用层。如果没有领域服务，应用层将会直接调用领域对象完成本属于领域服务的操作。这样领域层可能会把一部分领域知识泄露到应用层，因为应用层需要了解每个领域对象的业务功能，具有哪些信息，以及它可能会与哪些其他领域对象交互，怎么交互等一系列领域知识。

8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in 3-tier Architecture

- Web 服务软件的三层逻辑架构
  - ✧ 持久层 (Persistent Layer/Data Access Layer/DAL)
    - ◌ 持久层指把领域模型保存到数据库中。程序代码是面向对象风格的，而数据库一般是关系型的数据库，所以需要应用数据映像器，才能将数据保存到数据库中。例如采用行数据入口法 (Row Data Gateway)、表数据入口法 (Table Data Gateway) 或者把领域层和持久层合二为一变成活动记录 (Active Record) 的方式。
  - ✧ 基础设施层为其他层提供通用的技术能力、提供层间通信能力、为领域层实现持久化机制。基础设施层通过架构和框架来支持其他层的技术需求。

8.1 Overview

8.1.2 Web Site Architecture

- **Software: in 3-tier Architecture**
  - Web 服务软件的三层逻辑架构
    - 关于领域层逻辑的例：在 ATM 机上取款的业务流程。
      - 流程：客户插入银行卡，输入密码，输入取款金额，确定，取回现金，ATM 吐出一个交易凭条，客户取回银行卡。
      - 上述过程中，银行卡在 ATM 机器里完成取款金额从帐户上划拨的过程是一个领域逻辑，因为取款在银行业务中是一个明确的领域概念。ATM 机吐出一个交易凭条并不是领域逻辑，而仅是一个应用逻辑，因为吐出交易凭条不是银行中一个明确的领域概念，而只是一种技术手段，比如也可以不吐交易凭条而代之发送一条提醒短信。如果要求取款后必须吐出交易凭条，也即吐出交易凭条已经和取款紧密结合，那么吐出交易凭条就成为了领域逻辑的一部分。

8.1 Overview

8.1.2 Web Site Architecture

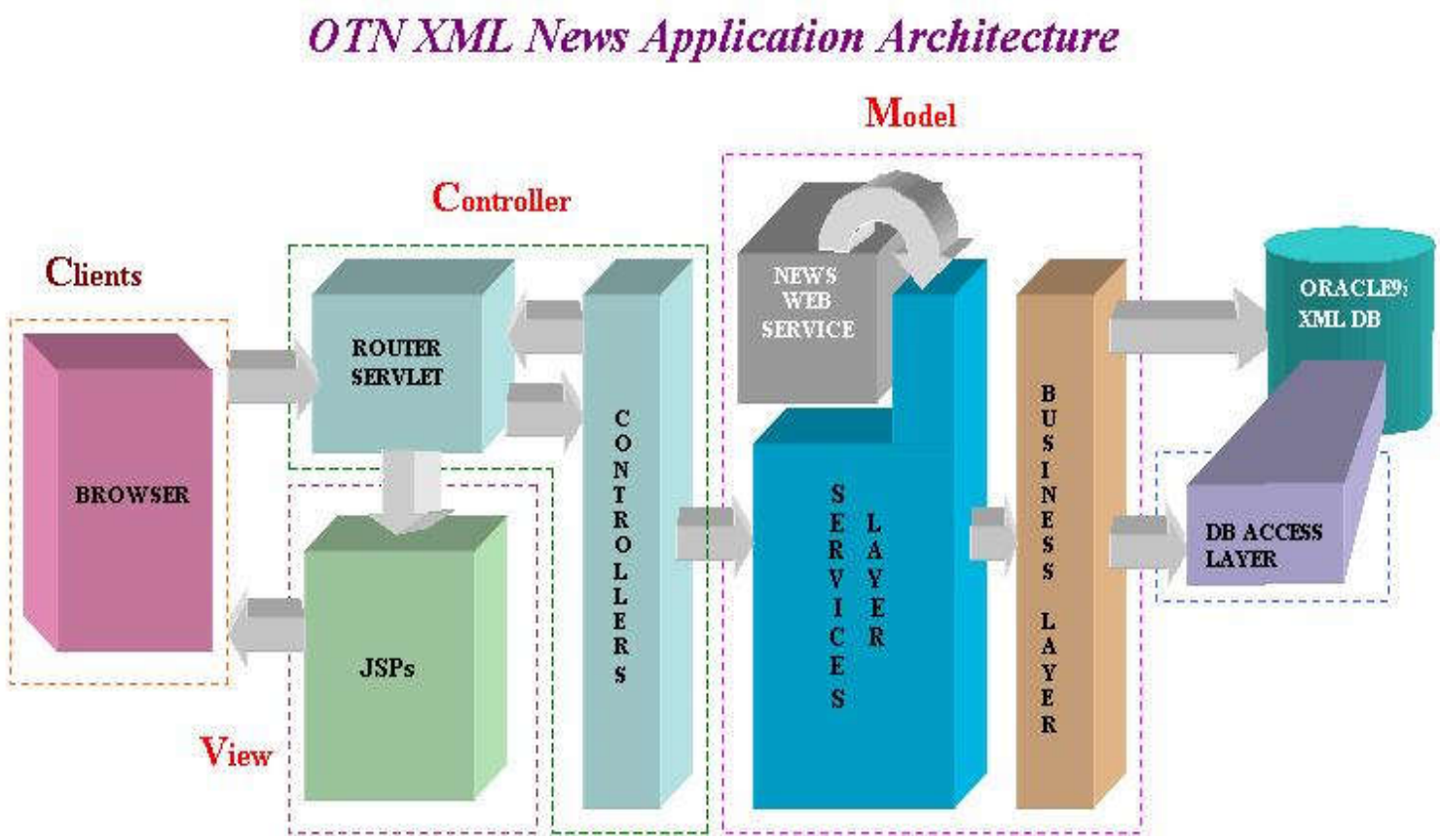
❑ Software: in 3-tier Architecture

- Web 服务软件的三层逻辑架构
  - ✧ 关于领域层服务的例：网上转帐的业务流程。
    - ◌ 应用层服务：
      - ① 获取输入 (如一个XML请求)
      - ② 发送消息给领域层服务，要求其实现转帐的业务逻辑
      - ③ 领域层服务处理成功，调用基础层服务向用户发送通知
    - ◌ 领域层服务：
      - ① 获取源帐号和目标帐号，分别通知源帐号和目标帐号进行扣除金额和增加金额的操作
      - ② 提供返回结果给应用层
    - ◌ 基础层服务：
      - ① 按照应用层的请求，向用户发送通知

8.1 Overview

8.1.2 Web Site Architecture

- **Software: in MVC (Model-View-Controller)**



8.1 Overview

8.1.2 Web Site Architecture

❑ Software: in MVC (Model-View-Controller)

- **Model–view–controller** is an architectural pattern commonly used for developing *user interfaces* that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples (解耦) these major components allowing for efficient code reuse and parallel development.
- Traditionally used for desktop graphical user interfaces (GUIs), this architecture has become popular for designing web applications and even mobile, desktop and other clients. Popular programming languages like Java, C#, Ruby, PHP have MVC frameworks that are used in web application development straight out of the box.
- See also:
  - OTN, Oracle Technology Network, Oracle® XML DB Developer's Guide - Part Number B28369-04
  - http://zh.wikipedia.org/wiki/MVC

8.1 Overview

8.1.3 Electronic Commerce Architecture

❑ Requirement

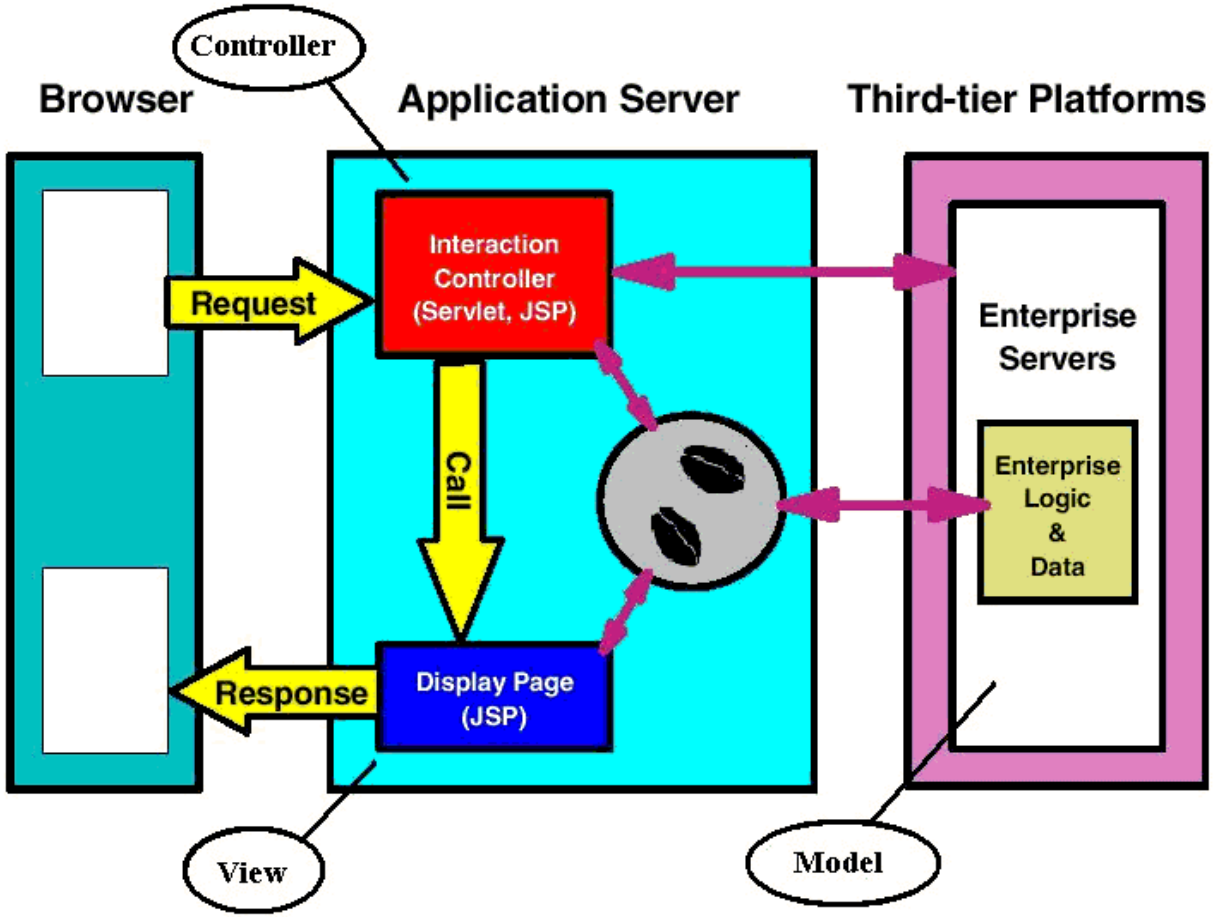
- Conveniences
- High security
- High efficiency
- High load
- High synchronization

  So…
- Online
- Server centered
- B/S
- Standard
- High scalability

8.1 Overview

8.1.3 Electronic Commerce Architecture

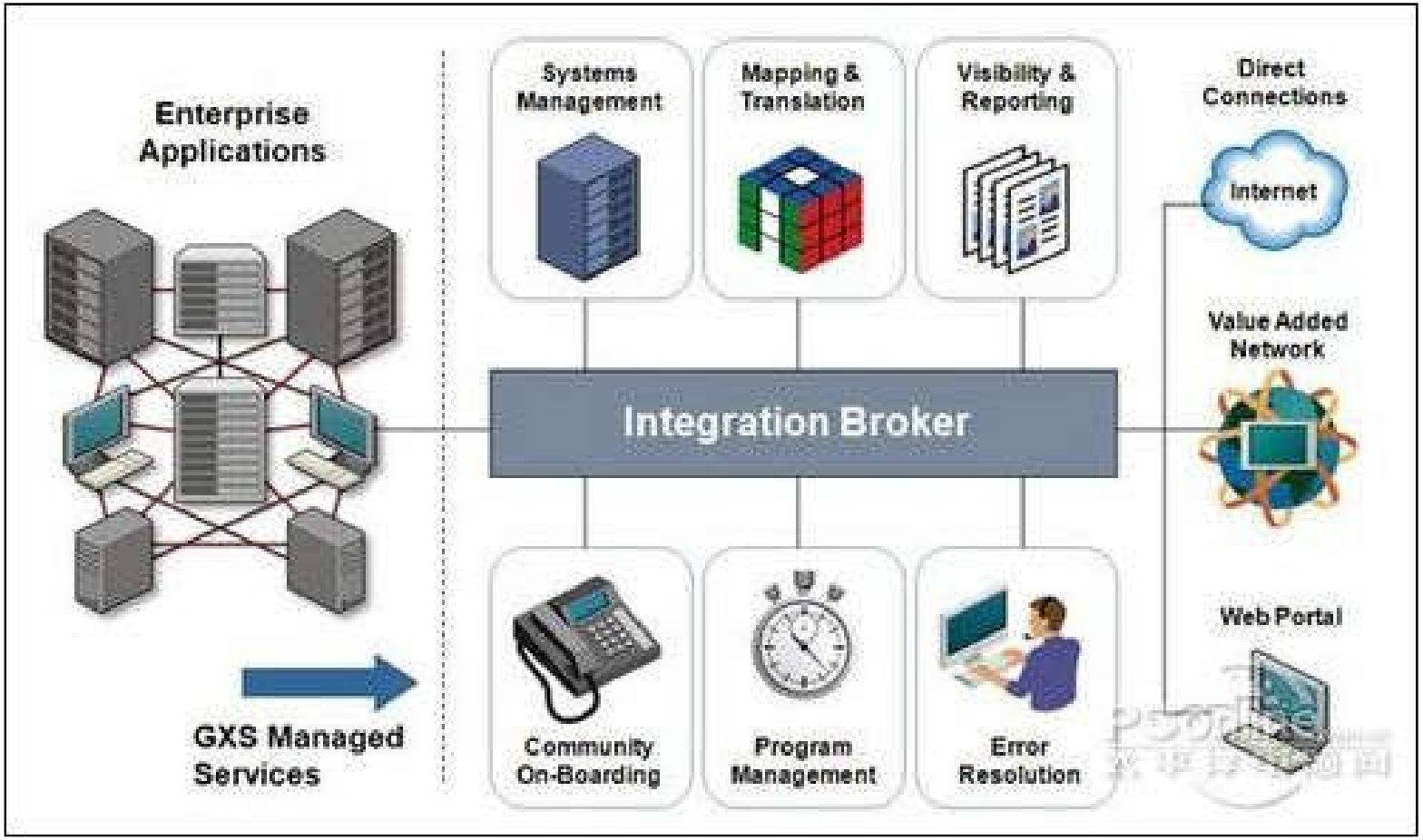
- **EJB (Enterprise JavaBean) in MVC**



8.1 Overview

8.1.3 Electronic Commerce Architecture

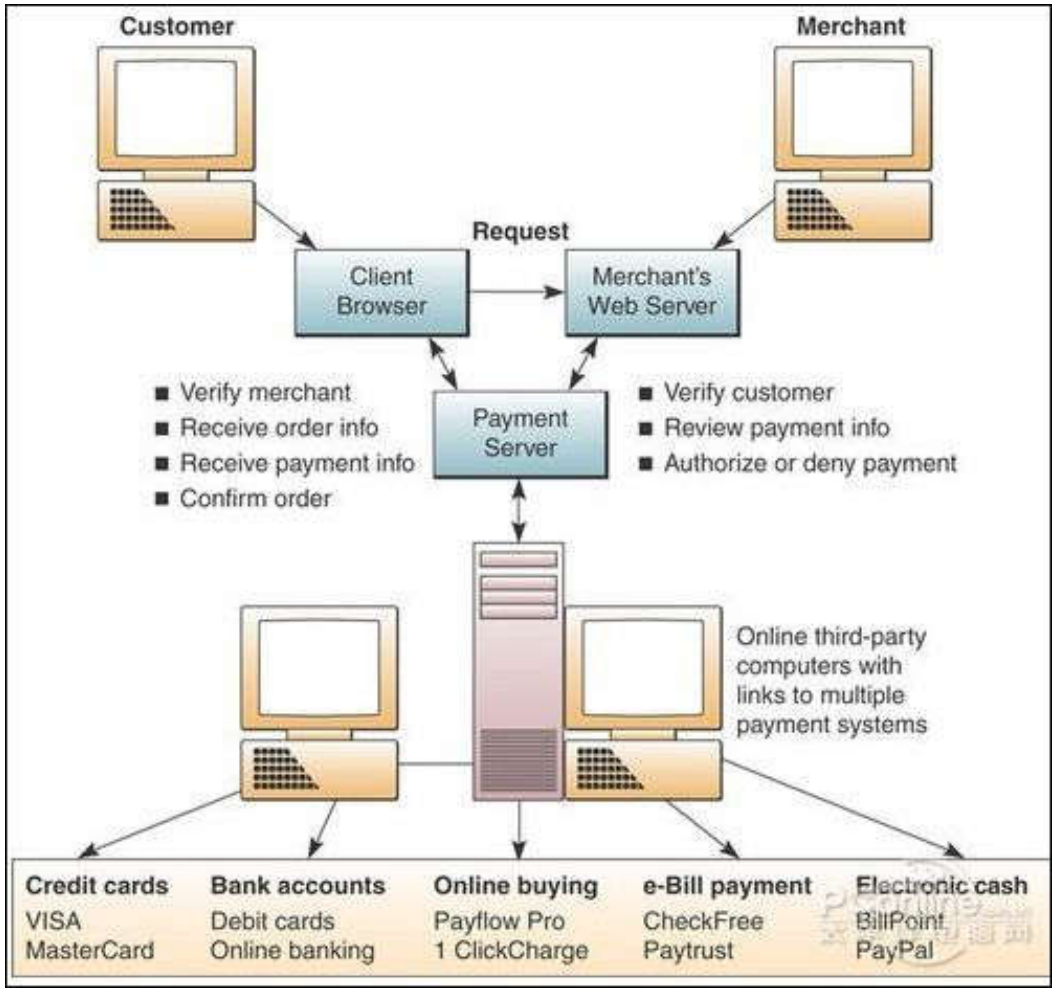
- **B2C (Amazon)**



8.1 Overview

8.1.3 Electronic Commerce Architecture

- C2C (Taobao)



淘宝网

8.1 Overview

8.1.3 Electronic Commerce Architecture

- **C2C (Taobao)**
  - High Scalability
    - Distributed computing
    - Load balance
    - Stateless application
    - Cache
    - Application division
    - Read/write division
    - Database division
    - Table division
    - Asynchronous communication
    - Unstructured DS
    - Monitoring and early Warning system
    - Others

8.1 Overview

8.1.3 Electronic Commerce Architecture

❑ C2C (Taobao)

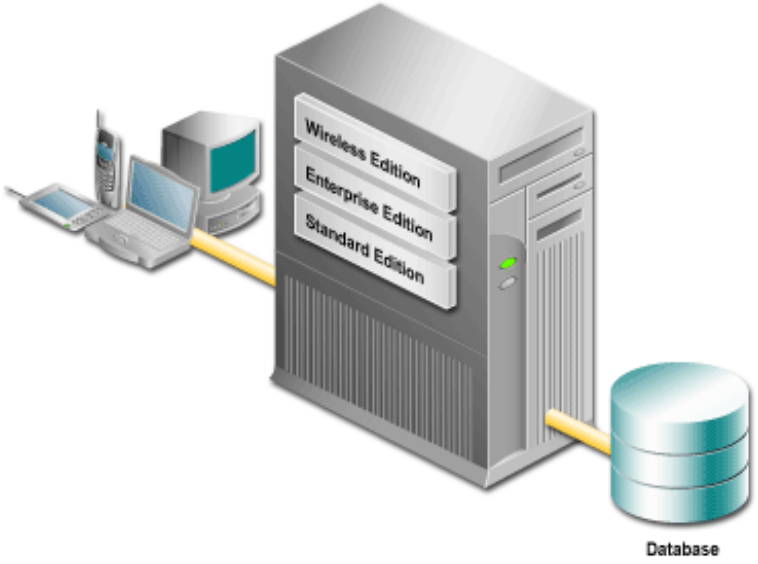
- WebX
  - ✧ WebX is some common module between Page and Service (action, controller, screen)
  - ✧ Common website: Page - Servlet - Service - DAO – DB
  - ✧ Taobao: Page - WebX - Service - DAO – DB
- Technical Environment
  - ✧ Hardware: PC server
  - ✧ OS: Linux, FreeBSD
  - ✧ Database: MySQL, Oracle 10g
  - ✧ Programming Language: C/C++, Java
  - ✧ Server: JBoss Application Server (JEE)
  - ✧ WebX: A common web application framework built on Java Servlet API

8.1 Overview

8.1.4 Apache & IIS

❑ HTTP Server

- What is HTTP Server
  - ✧ An HTTP server is a software/process running on a server machine, following HTTP protocol, responding REQUEST from client and returning RESPONSE.
  - ✧ It is used to deal with HTML files.



8.1 Overview

8.1.4 Apache & IIS

❑ Apache HTTP Web Server

- Apache Server
  - ✧ Running on many kinds of Unix-Like System: UNIX, GNU, FreeBSD, Linux, Solaris, Novell NetWare, Amiga OS, Mac OS X, Microsoft Windows, OS/2, TPF
  - ✧ Open Source
  - ✧ Simple, Efficient, Stable
  - ✧ Process based (unsuitable for multiprocessor environment)
  - ✧ HTTP 1.1, Perl, Proxy, Common Gateway, SSI, SSL, FastCGI supported
  - ✧ HTTP/2 supported
  - ✧ The latest version: 2.4.37 (2018.10.23)

8.1 Overview

8.1.4 Apache & IIS

❑ Apache HTTP Web Server

- History of Apache

Original author(s)	*Robert McCool*
Developer(s)	Apache Software Foundation (ASF)
Initial release	1995
Stable release	2.4.37 (Oct. 23, 2018)
Development status	Active
Written in	C, XML
Operating system	Unix-like, Windows
Type	Web server
License	Apache License 2.0
Website	httpd.apache.org
Repository	svn.apache.org/repos/asf/httpd/httpd

8.1 Overview

8.1.4 Apache & IIS

❑ Apache HTTP Web Server

- Tomcat (Servlet Container, for jsp)

8.1 Overview

8.1.4 Apache & IIS

❑ Apache HTTP Web Server

- Apache 服务器
  - ✧ Apache 源于 NCSA httpd 服务器，经过多次修改，逐步扩充到各种 Unix 系统中，成为主流 Web 服务器软件，也是使用排名第一的 Web 服务器软件。
  - ✧ Apache 源代码开放，可以运行在几乎所有广泛使用的操作系统平台 (Unix、Windows、Linux) 上，其特点是简单、速度快、性能稳定、可移植，并可做代理服务器使用。
  - ✧ Apache 有多种产品，可以支持 SSL 技术，支持多个虚拟主机。Apache 是以进程为基础的结构，进程要比线程消耗更多的系统开支，不太适合于多处理器环境，因此，在一个 Apache Web 站点扩容时，通常是增加服务器或扩充集群节点而不是增加处理器。

8.1 Overview

8.1.4 Apache & IIS

❑ Apache HTTP Web Server

- Apache web 服务器软件拥有以下特性：
  - ✧ 支持 HTTP/1.1、HTTP/2 通信协议
  - ✧ 拥有简单而强有力的基于文件的配置过程
  - ✧ 支持通用网关接口
  - ✧ 支持基于 IP 和基于域名的虚拟主机
  - ✧ 支持多种方式的 HTTP 认证
  - ✧ 集成 Perl 处理模块
  - ✧ 集成代理服务器模块
  - ✧ 支持实时监视服务器状态和定制服务器日志
  - ✧ 支持服务器端包含指令 (SSI)
  - ✧ 支持安全 Socket 层 (SSL)
  - ✧ 提供用户会话过程的跟踪
  - ✧ 支持 FastCGI
  - ✧ 通过第三方模块可以支持 Java Servlets (Tomcat)

8.1 Overview

8.1.4 Apache & IIS

❑ IIS Web Server

- IIS Server
  - ✧ IIS (Internet Information Services) is a Web service module of Microsoft, including Web server, FTP server, NNTP (Net News Transport Protocol) server and SMTP server.
  - ✧ Running on windows only with HTTP, HTTPS, FTP, FTPS, SMTP and NNTP Supported, and with ASP (Active Server Pages), Java, VBscript supported.
  - ✧ Extend function: FRONTPAGE, INDEX SERVER, NET SHOW.
  - ✧ The latest version: IIS 10.0 Version 1809, shipped with Windows 10 October Update 2018 and Window Server 2019.

8.1 Overview

8.1.4 Apache & IIS

❑ IIS Web Server

- IIS 服务器
  - ✧ IIS 是由微软开发的一种 Web 服务组件，其中包括 Web 服务器、FTP 服务器、NNTP 服务器和 SMTP 服务器，分别用于网页浏览、文件传输、新闻服务和邮件发送等服务，它使得在网络 (包括互联网和局域网) 上发布信息成了一件很容易的事。IIS 是随 Window Server 一起提供的文件和应用程序服务器，是在 Window Server 上建立 Internet服务器的基本组件。它与 Window Server 完全集成，允许使用 Windows Server 内置的安全性以及 NTFS 文件系统建立强大灵活的 Internet / Intranet 站点。

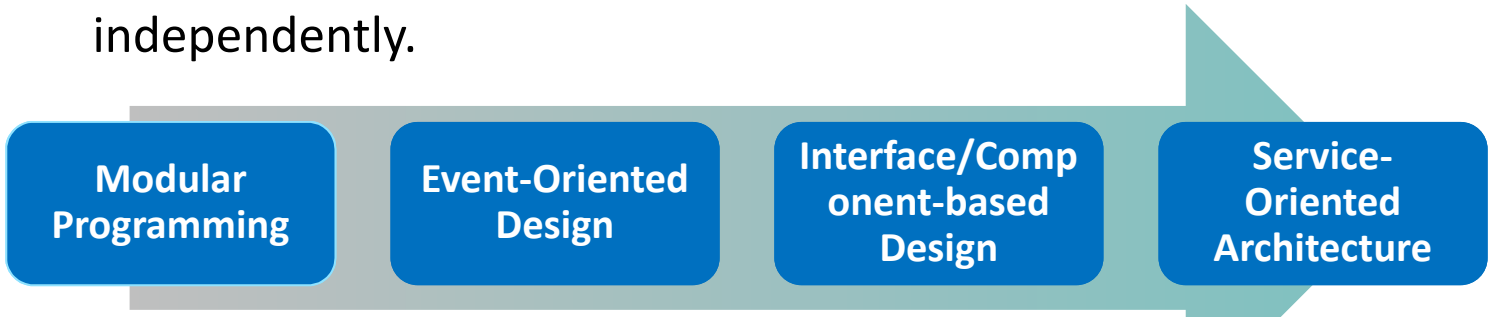
Outline

- 8.1 Overview
  - C/S and B/S Models
  - Web Site Architecture
  - Electronic Commerce Architecture
  - Apache & IIS
- **8.2 Web Services**
  - SOA
  - Web Services
  - SOAP
- 8.3 Web Security Primer
  - Web Security - Beginning
  - Web Server Vulnerabilities
  - Web Services Secure Model
  - Prevention of Malicious Codes
  - SSL/HTTPS

8.2 Web Services

8.2.1 SOA

- **SOA** (Service-Oriented Architecture)
  - What is SOA
    - SOA is a flexible set of design principles used during the phases of systems development and integration in computing. A system based on an SOA will package functionality as a suite of interoperable services (具备互操作性的服务) that can be used within multiple, separate systems from several business domains.
    - The basic principles of SOA are independent of vendors, products and technologies. A service is a discrete unit of functionality that can be accessed remotely and acted upon and updated independently.



The Evolution of SOA

8.2 Web Services

8.2.1 SOA

❑ SOA

- SOA Baseline Foundation
  - ✧ Services generally adhere to the following principles of service-orientation, forming the baseline foundation for SOA.
    - ◌ Encapsulation (封装)
    - ◌ Autonomy (自治)
    - ◌ Formal contract (服务规约)
    - ◌ Loose coupling (松耦合)
    - ◌ Abstraction (抽象)
    - ◌ Composability (可组构性)
    - ◌ Reusability (可重用性)
    - ◌ Statelessness (状态无关性/无状态架构)
    - ◌ Discoverability (可发现性)

8.2 Web Services

8.2.1 SOA

❑ SOA

- Autonomy (自治)
  - ✧ Services are autonomous — The logic governed by a service resides within an explicit boundary. The service has control within this boundary, and is not dependent on other services for it to execute its governance.
- Formal Contract (服务规约)
  - ✧ Services share a formal contract — In order for services to interact, they need not share anything but a collection of published metadata that describes each service and defines the terms of information exchange.
- Loose Coupling (松耦合)
  - ✧ Services are loosely coupled — Dependencies between the underlying logic of a service and its consumers are limited to conformance (一致性) of the service contract.

8.2 Web Services

8.2.1 SOA

❑ SOA

- Abstraction (抽象)
  - ✧ Services abstract underlying logic — Underlying logic, beyond what is expressed in the service contract metadata, is invisible to the outside world.
- Composability (可组构性)
  - ✧ Services are composable — Services may compose others, allowing logic to be represented at different levels of granularity (粒度). This promotes reusability and the creation of service abstraction layers.
- Reusability (可重用性)
  - ✧ Services are reusable — Regardless of whether immediate reuse opportunities exist, services are designed to support potential reuse.

8.2 Web Services

8.2.1 SOA

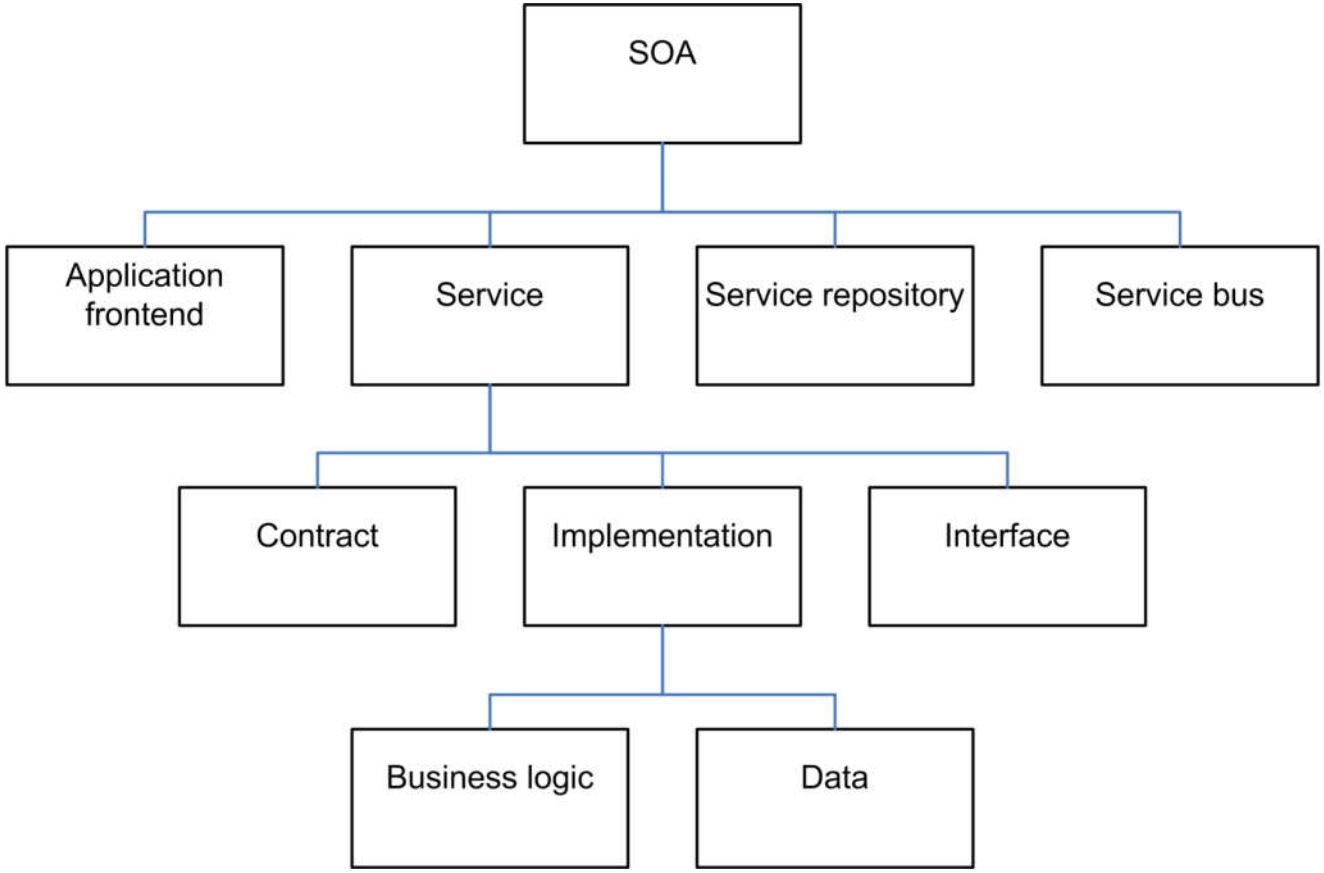
❑ SOA

- Statelessness (状态无关性/无状态架构)
  - ✧ Services are stateless — Services should be designed to maximize statelessness even if that means deferring state management elsewhere.
- Discoverability (可发现性)
  - ✧ Services are discoverable — Services should allow their descriptions to be discovered and understood by humans and service requestors that may be able to make use of their logic.

8.2 Web Services

8.2.1 SOA

- **SOA**

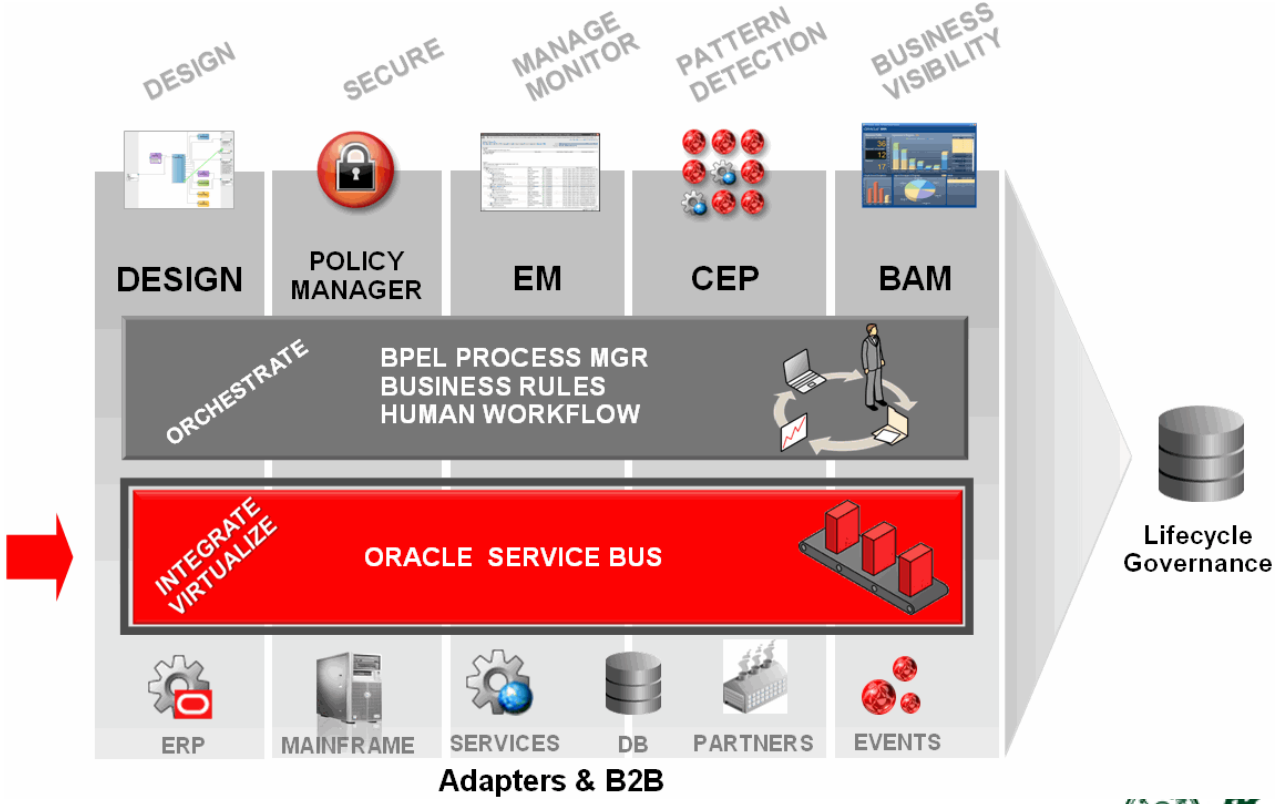


8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- There are no industry standards relating to the exact composition of an SOA. Many industry sources have published their own principles.
- Oracle's conceptual architecture for SOA



8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- Oracle's conceptual architecture for SOA
  - ✧ Oracle's SOA conceptual architecture is separated into life cycle stages: *Design*, *Secure*, *Manage/Monitor*, *Pattern Detection*, and *Business Visibility*.
  - ✧ Along the bottom of the diagram are examples of key elements in an SOA environment: *ERP*, *Mainframe*, *Services*, *Database*, *Partners*, and *Events*.
  - ✧ Running across all stages are two boxes: Orchestrate (策划) and Integrate/Virtualize.
    - ◌ The Orchestrate box includes BPEL Process Manager, Business Rules, and Human Workflow.
      - BPEL: *Business Process Execution Language*
    - ◌ The Integrate/Virtualize box is Oracle Service Bus.

8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

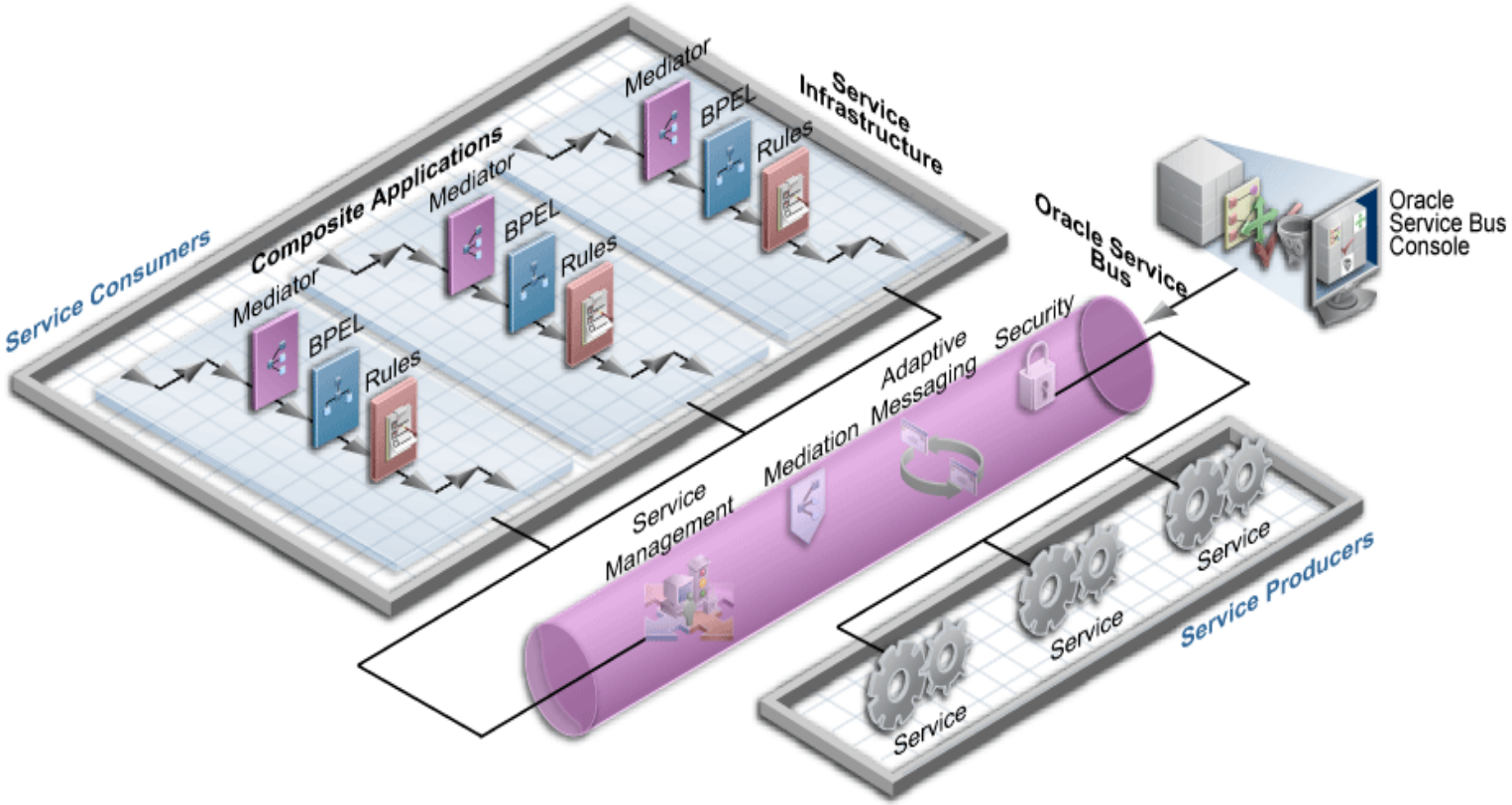
- Oracle's conceptual architecture for SOA
  - ✧ BPEL is the standard for assembling a set of discrete services into an end-to-end process flow, radically reducing the cost and complexity of process integration initiatives.
  - ✧ Oracle BPEL Process Manager offers a comprehensive and easy-to-use infrastructure for creating, deploying and managing BPEL business processes.

8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- Service Bus Intermediary



8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

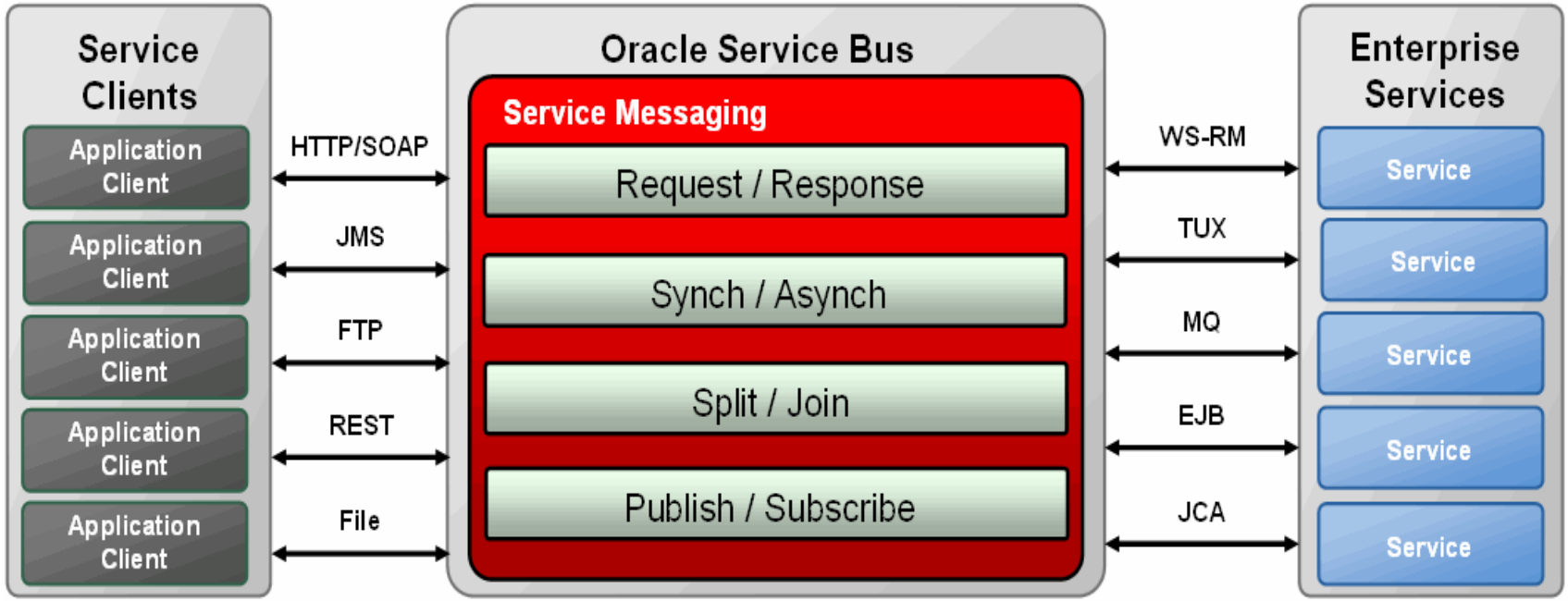
- Service Bus Intermediary
  - ✧ Oracle Service Bus, in the center, serves as an intermediary between Service Consumers and Service Producers.
  - ✧ The Oracle Service Bus layer includes Service Management, Mediation (调节), Adaptive Messaging, Security, and service management tooling, such as the Oracle Service Bus Console.

8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- Service Bus Intermediary
  - ✧ Adaptive Messaging in Oracle Service Bus



8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

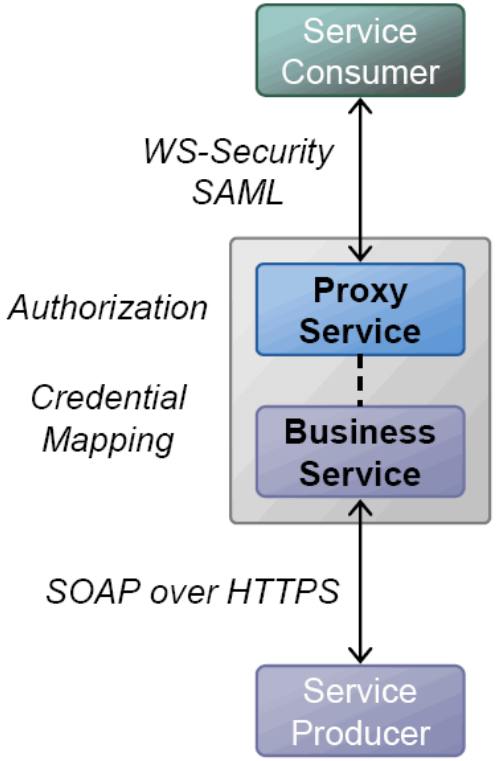
- Service Bus Intermediary
  - ✧ Adaptive Messaging in Oracle Service Bus
    - ◌ Oracle Service Bus node, in the middle, contains Service Messaging items: Request/Response, Synch/Asynch, Split-Join, and Publish/Subscribe.
    - ◌ A group of Service Clients/Application Clients sits to the left and communicate with the bus using different protocols: HTTP/SOAP, JMS, FTP, REST, and File.
    - ◌ On the right are Enterprise Services that communicate with the bus using different protocols: WS-RM, TUX, MQ, EJB, and JCA.

8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- Optimized Pluggable Security Layer over HTTPS
  - ✧ Security features and support at different points in a message life cycle include:
  - ✧ **Service Consumer**:
    - ◌ Communicates with Oracle Service Bus using WS-Security and SAML (Security Assertion Markup Language).
  - ✧ **Oracle Service Bus:**
    - ◌ With a proxy and business service that leverage pluggable Authorization and Credential Mapping providers.
  - ✧ **Service Producer:**
    - ◌ Communicates with Oracle Service Bus using SOAP over HTTPS.



8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- Optimized Pluggable Security Layer over HTTPS
  - ✧ Based on Oracle Platform Security Services and Oracle WebLogic security framework, Oracle Service Bus ensures service security at all levels. A comprehensive set of components for built-in security gives customers significant flexibility and choice.
  - ✧ Users can also plug in custom or third-party security components. Built-in capabilities allow flexibility in implementation by enabling security at all levels.

8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- Optimized Pluggable Security Layer over HTTPS
  - ✧ *Example*:
    - ◌ Transport Security
      - SSL/Basic authentication and custom security credentials
    - ◌ Message Security
      - WS-Policy/WS-Security, SAML, User ID/Password, X509, Signing & Encryption, and custom security credentials
    - ◌ Console Security
      - Web Single-Sign-On and role-based access
    - ◌ Policy Security
      - WS-Security, Oracle Web Services Manager, and WS-Policy

8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- Security features Provided by Oracle Service Bus:
  - ✧ Integration with Oracle Web Services Manager
  - ✧ Authentication, encryption and decryption, and digital signatures as defined in the Web Services Security (WS-Security) specification
  - ✧ Uses SSL to support traditional transport-level security for HTTP and JMS transport protocols
  - ✧ One-way and two-way certificate based authentication
  - ✧ HTTP basic authentication
  - ✧ Encrypt and export of resources (such as service accounts, service key providers, UDDI registries, SMTP providers, and JNDI providers) that contain username and passwords

8.2 Web Services

8.2.1 SOA

❑ Oracle's SOA

- Security features Provided by Oracle Service Bus:
  - ✧ Create service accounts and service key providers within a session, and add the user name, password, and credential alias binding within the same session.
  - ✧ Configure a service account to pass through user ID and password credentials or map the user to a new user ID and password supplied to a business service
  - ✧ Client-specified custom authentication credentials for both transport- and message-level inbound (呼入) requests

8.2 Web Services

8.2.2 Web Services

❑ Web Service

- What is a Web Service
  - ✧ A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.

    - W3C, Web Services Glossary

    - W3C: World Wide Web Consortium

8.2 Web Services

8.2.2 Web Services

❑ Web Service

- Three Patterns (Software Architectural Styles) of Web Services
  - ✧ RPC (Remote Procedure Call, 远程过程调用, 1981)
  - ✧ SOA (Service-Oriented Architecture, 面向服务架构, 1998)
  - ✧ REST (REpresentational State Transfer, 表示状态转移, 2000)

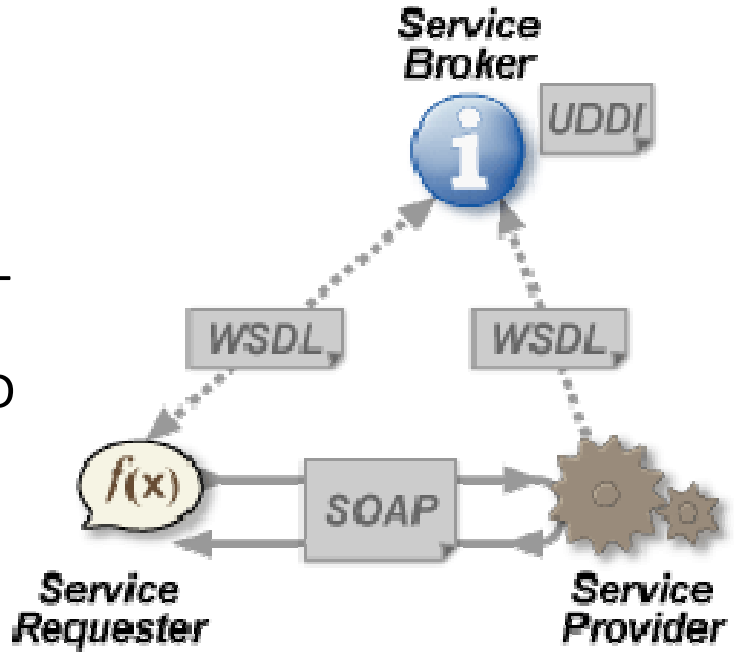
  - ✧ See also: *Michael Mahemoff*, Ajax Design Patterns, O'Reilly 2006

8.2 Web Services

8.2.2 Web Services

❑ Web Service

- Architecture for Web Services
  - ✧ The service provider sends a WSDL file to UDDI.
  - ✧ The service requester contacts UDDI to find out who is the provider for the data it needs, and then it contacts the service provider using the SOAP protocol.
  - ✧ The service provider validates the service request and sends structured data in an XML file, using the SOAP protocol. This XML file would be validated again by the service requester using an XSD file.



8.2 Web Services

8.2.2 Web Services

❑ Web Service

- Web 服务
  - ✧ W3C：一个 Web 服务是一个软件系统，为了支持跨网络的机器间互操作交互而设计。
  - ✧ Web 服务是一种 Web 应用程序，发布在网络 (通常为 Web) 中，使用者通过 Web 调用其 API 来实现其功能。
  - ✧ Web 服务是一种跨编程语言和跨操作系统平台的远程调用技术，它希望不同的系统之间能够用“软件-软件对话”的方式相互调用，在软件应用、网站和各种设备之间进行协调，实现“基于 Web 无缝集成”的目标。
  - ✧ Web 服务是基于网络的，自包含、自描述、分布式的模块化组件。它执行特定的任务，遵守具体的技术规范，这些规范使得 Web 服务能与其他兼容的组件进行互操作。

8.2 Web Services

8.2.2 Web Services

- **Web Service**
  - Web 服务
    - Web 服务平台包含了一套协议标准，用于沟通不同平台、编程语言和组件模型中的不同类型系统。
      - 构造 Web 服务平台的三个技术范畴
        - Web Service = XML+XSD, SOAP and WSDL.

8.2 Web Services

8.2.2 Web Services

❑ Protocols for Web Services

- Web 服务协议

(1) XML (eXtensible Markup Language) 和 XSD (XML Schema Definition)

- 可扩展标记语言 XML 由 W3C 创建，是 Web 服务平台中表示数据的基本格式。XML 易于建立，易于分析，与平台无关，又与厂商无关。
- W3C 制定并建议的 XML Schema XSD 定义了一套标准的数据类型，并给出了一种语言来扩展这套数据类型。
- Web 服务平台采用 XSD 作为数据类型系统。当某种语言 (如 VB.NET 或 C#) 被用来构造一个 Web 服务时，为了符合 Web 服务标准，所有使用的数据类型都必须转换为 XSD 类型。如想让它使用在不同平台和不同软件的不同组织间传递，还需要对其包装。定义这种包装的就是一种协议，如 SOAP。

8.2 Web Services

8.2.2 Web Services

❑ Protocols for Web Services

- Web 服务协议

(2) SOAP (Simple Object Access Protocol)

- 简单对象访问协议 SOAP 是用于交换 XML 编码信息的轻量级协议。
- XML-envelope 为描述信息内容和如何处理内容定义了框架
  - 将程序对象编码成为 XML 对象的规则
  - 执行远程过程调用 (RPC) 的约定。
- SOAP 可以运行在任何其他传输协议上。例如，可以使用 SMTP 因特网电子邮件协议来传递 SOAP 消息。

8.2.2 Web Services

❑ Web Services Protocols

- Web 服务协议

(3) WSDL (Web Services Description Language)

- Web 服务描述语言 WSDL 用机器能阅读的方式提供基于 XML 的语言的正式描述文档，用于描述 Web 服务及其函数、参数和返回值。因为是基于 XML 的，所以 WSDL 既是机器可阅读的，又是人可阅读的。

(4) UDDI (Universal Description, Discovery, and Integration)

- UDDI 是一套基于 Web 的、分布式的、为 Web 服务提供的信息注册中心的实现标准规范，同时也包含一组使企业能将自身提供的 Web 服务注册，以使别的企业能够发现的访问协议的实现标准。

8.2 Web Services

8.2.2 Web Services

❑ Web Services Protocols

- Web 服务协议

(5) RPC (Remote Procedure Call) and Message Passing

- Web 服务本身实现的是应用程序间的通信，包括远程过程调用 RPC 和消息传递。
- 使用 RPC 的时候，客户端调用服务器上的远程过程，通常方式为实例化一个远程对象并调用其方法和属性。
- RPC 系统试图达到一种位置上的透明性：远端服务器暴露出远程对象的接口，客户端在本地使用对象接口而无需了解对象的位置，从而隐藏了底层信息。

8.2 Web Services

8.2.2 Web Services

❑ Web Services Protocols

- Web Services Protocol Stack
  - ✧ A **web service protocol stack** is used to define, locate, implement, and make Web services interact with each other. A Web service protocol stack typically stacks four protocols:
  - ✧ **(Service) Transport Protocol**: responsible for transporting messages between network applications and includes protocols such as HTTP, SMTP, FTP, as well as the more recent Blocks Extensible Exchange Protocol (BEEP).
  - ✧ **(XML) Messaging Protocol**: responsible for encoding messages in a common XML format so that they can be understood at either end of a network connection. Currently, this area includes such protocols as XML-RPC, WS-Addressing, and SOAP.
  - ✧ **(Service) Description Protocol**: used for describing the public interface to a specific Web service. The WSDL interface format is typically used for this purpose.

8.2 Web Services

8.2.2 Web Services

❑ Web Services Protocols

- Web Services Protocol Stack
  - ✧ **(Service) Discovery Protocol**: centralizes services into a common registry so that network Web services can publish their location and description, and makes it easy to discover what services are available on the network. UDDI was intended for this purpose, but it has not been widely adopted.
  - ✧ The Web service protocol stack also includes a whole range of recently defined protocols: BPEL, SOAP-DSIG.

8.2 Web Services

8.2.2 Web Services

❑ Web Services Protocols

- Web Services Protocol Stack

Tool	Layer	Business Issues		
WSFL	Service Flow	Security	Management	Quality of Service
Static ->UDDI	Service Discovery			
Direct -->UDDI	Service Publication			
WSDL	Service Description: --Service Impl --Service Interface			
SOAP	XML-Based Messaging			
XML Schema	Data Modal			
XML	Data Presentation			
HTTP, FTP, SMTP, MQ	Transport			

8.2 Web Services

8.2.2 Web Services

❑ Web Services Development

- Web 服务的支持软件
  - ✧ 目前 Web 服务的主流实现技术是 .NET 和 Java，并且两种实现方法可以互操作；可以看到使用微软、IBM、SUN、Borland 等不同厂商的 Web 服务构建工具建立的 Web 服务应用。
  - ✧ 微软的 .NET
    - ◌ .NET 平台延续了微软一贯的编程风格，增加了许多支持 Web 服务的关键性技术，使得 .NET 在操作的简单性和执行的稳定性、高效性上达到了一个很好的结合。用于 Web 服务开发的主要工具是 ASP.NET，实行代码和 HTML 分离，支持非脚本语言如 C#。

8.2 Web Services

8.2.2 Web Services

❑ Web Services Development

- Web 服务的支持软件
  - ✧ IBM 的 WebSphere
    - ◌ 包括 DB2、Lotus、Tivoli 和 WebSphere 在内的所有 IBM 软件都实现了对 SOAP、WSDL、UDDI、Linux、XML、J2EE 等开放技术和标准的全面支持。
    - ◌ WebSphere 软件平台及开发工具包括 WSAD (WebSphere Studio Application Developer)，基于 J2EE、XML 和 Web 服务等开放标准，并具备 IBM 在可靠性、扩展性和安全性上的主要优势。WebSphere 是 IBM 在 Web Services 策略中的核心平台，它支持所有开发、发布、部署 Web 服务所必需的开放标准和技术，包括 UDDI，SOAP，J2EE，WSDL 以及对 XML 技术集成的增强，在全球拥有众多用户。

8.2 Web Services

8.2.2 Web Services

❑ Web Services Development

- Web 服务的支持软件
  - ✧ Borland 的 Jbuilder
    - ◌ 用户可以使用 Borland Web Services Kit for Java 和 Borland JBuilder MobileSet 3 更快捷地开发 Web 服务和无线应用。开发者能够在同一个开发环境中轻松地创建和集成 Web 服务。

8.2 Web Services

8.2.2 Web Services

❑ Web Services Development

- Web 服务的开发平台
  - ✧ Eclipse Foundation
    - ◌ Eclipse, IDE, Open source
  - ✧ Sun Microsystems
    - ◌ NetBeans, IDE, Open source
  - ✧ Jet Brains
    - ◌ IntelliJ IDEA
  - ✧ IBM
    - ◌ WSAD (Web Sphare Application Developer), Java IDE
  - ✧ Microsoft
    - ◌ Visual J++, Java IDE
  - ✧ Borland
    - ◌ JBuilder, Java IDE

8.2 Web Services

8.2.3 SOAP

❑ What is SOAP

- SOAP is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks.
- SOAP allows processes running on disparate operating systems (such as Windows and Linux) to communicate using Extensible Markup Language (XML). Since Web protocols like HTTP are installed and running on all operating systems, SOAP allows clients to invoke web services and receive responses independent of language and platforms.
- SOAP provides the Messaging Protocol layer of a web services protocol stack for web services. It is an XML-based protocol consisting of three parts:
  - ✧ an envelope, which defines the message structure and how to process it
  - ✧ a set of encoding rules for expressing instances of application-defined datatypes
  - ✧ a convention for representing procedure calls and responses

8.2 Web Services

8.2.3 SOAP

❑ What is SOAP

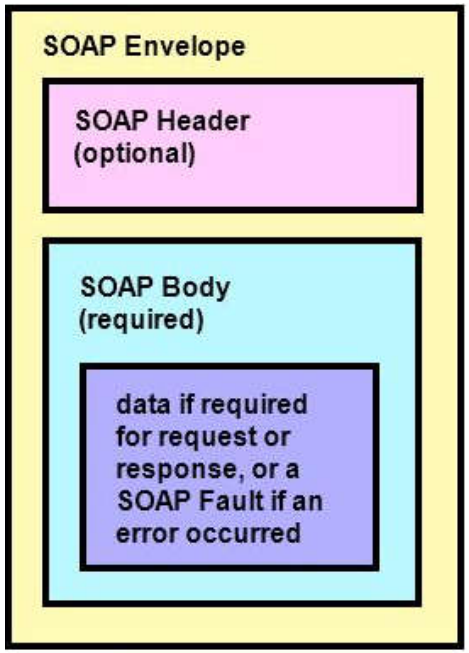
- SOAP has three major characteristics:
  - ✧ Extensibility (可扩展性)
    - ◌ Security and WS-Addressing are among the extensions under development.
  - ✧ Neutrality (中立性)
    - ◌ SOAP can operate over any protocol such as HTTP, SMTP, TCP, UDP, or JMS.
  - ✧ Independence (独立性)
    - ◌ SOAP allows for any programming model.

8.2 Web Services

8.2.3 SOAP

❑ What is SOAP

- An SOAP message is an ordinary XML document includes:
  - ✧ Envelope: Identifies the XML document as a SOAP message.
  - ✧ Header: Contains header information.
  - ✧ Body: Contains call, and response information.
    - ◌ Fault: Provides information about errors that occurred while processing the message.



8.2 Web Services

8.2.3 SOAP

❑ What is SOAP

- HTTP 协议作为用于 Web 浏览器和 Web 服务器之间通信的主要应用协议，其最大的不足之处在于只具备简单的请求和发送等传输命令，比如：Get、Post、Put，作用也仅仅主要体现在对数据的浏览功能等。
- 简单对象访问协议 SOAP 产生于1998年，用于满足应用程序之间的数据共享和数据交换，其首要目的是解决不同平台和不同系统之间的应用程序之间的通信。XML 语言成为 SOAP 首选的大纲语言。

8.2 Web Services

8.2.3 SOAP

❑ SOAP Structure

- SOAP 的结构也称 SOAP 消息，包含三个主要元素

  Soap <Envelope>, Soap <Header>, Soap <Body>

  - ✧ Soap <Envelope>：消息根 (或消息标题)，必需项，用于告诉接收方此消息的基本情况。
  - ✧ Soap <Header>：消息头，可选项，若存在则必须是Envelope 的第一个子元素，用于传递一些辅助消息如身份验证，会话等。
  - ✧ Soap <Body>：消息体，必需项，一个 body 元素可以由多个子元素构成。
    - ◌ Soap < Fault>：错误报告，可选项，属于 body 元素中的特殊元素，当发送消息产生错误时使用这个元素来报告消息的错误信息。

Outline

- 8.1 Overview
  - C/S and B/S Models
  - Web Site Architecture
  - Electronic Commerce Architecture
  - Apache & IIS
- 8.2 Web Services
  - SOA
  - Web Services
  - SOAP
- **8.3 Web Security Primer**
  - Web Security - Beginning
  - Web Server Vulnerabilities
  - Web Services Secure Model
  - Prevention of Malicious Codes
  - SSL/HTTPS

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Why not Secure

- Web 的初始目的是提供快捷服务和直接访问，早期的 Web 没有考虑安全性问题。随着 Web 的广泛应用，Internet 中与 Web 相关的安全事故正成为目前所有信息安全事故的主要组成部分。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Why not Secure

- Web 安全问题的客观原因
  - ✧ Web 平台的复杂性
    - ◌ 操作系统漏洞
    - ◌ Web 服务器软件漏洞
    - ◌ 运行于 Web 服务器上的各种商业软件的漏洞
    - ◌ Web 应用程序自身的漏洞
    - ◌ 漏洞的修补
      - 第三方软件：补丁的发布远远滞后于黑客发现并利用漏洞 (0day)。
      - Web 应用程序：检测、修补应用漏洞费时费力。
      - 现有技术架构下，网站漏洞将长期存在。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

- **Why not Secure**
  - Web 安全问题的客观原因
    - 传统网络安全设备
      - 防火墙
        - 限制地址和端口访问
        - 验证和加固网络协议
      - 入侵检测
        - 基于网络层的数据包检查
      - 问题
        - Web/80 端口的保护
        - 应用数据的保护
        - 信息完整性的保护

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Why not Secure

- Web 安全问题的客观原因
  - ✧ 传统防护手段
    - ◌ 传统的安全设备如防火墙、安全网关、IDS /IPS、审计产品、终端防护产品等，作为网站整体安全策略中不可或缺的重要模块，其对网络、通信协议、操作系统、数据库等通用内容的防护效果比较有效。
    - ◌ 针对 Web 特定应用的脆弱性以及产生的安全问题是个性化的，这些传统的技术手段不能有效防范和检测网站遭受的特定威胁和攻击。
      - 例如：跨站脚本攻击、信息泄露、SQL 注入、越权操作、DDoS 攻击 。 。 。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Why not Secure

- Web 安全问题的主观原因
  - ✧ 密码管理不善
    - ◌ 合格密码的建议未得到重视。
    - ◌ 35%的人与其他人共享密码。
    - ◌ 67%的人用同一密码访问多个程序或服务。
  - ✧ 配置不安全
    - ◌ 权限管控不当；信息泄露。
  - ✧ 漏洞修补不及时
    - ◌ 漏洞扫描不足；第三方软件更新不及时。
    - ◌ 新的应用上线前的安全测试不足。
  - ✧ 访问不良网站
    - ◌ 不良网站是钓鱼、木马、间谍软件常驻地。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Aims of Web Security

- Web 安全的目标包括
  - ✧ 保护 Web 服务器及其数据的安全
  - ✧ 保护 Web 服务器和用户之间信息传递的安全
  - ✧ 保护终端用户计算机及其他接入因特网设备的安全

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Aims of Web Security

- 保护 Web 服务器及其数据的安全
  - ✧ Web 服务器的安全保护
    - ◌ 保证系统持续不断的稳定、可靠的运行，从而提供可靠的 Web 服务。
    - ◌ 有效隔离非法访问：未经授权不得访问服务器。
    - ◌ 严格管理权限：系统文件未经授权不得访问。
  - ✧ Web 服务器的数据安全保护
    - ◌ 防止存储在服务器里的数据和配置信息未经授权被窃取、篡改或删除。
    - ◌ 只允许授权用户访问 Web 发布的信息。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Aims of Web Security

- 保护 Web 服务器和用户之间信息传递的安全
  - ✧ 确保用户提供给 Web 服务器的信息 (用户名、密码、财务信息、访问的网页名等) 不被第三方所窃听、篡改和破坏。
  - ✧ 同样地保护从 Web 服务器端发送给用户的信息。
  - ✧ 保护用户和服务器之间的链路安全，避免被攻击者破坏。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Aims of Web Security

- 保护终端用户计算机及其他接入因特网设备的安全
  - ✧ 保护终端用户计算机的安全
    - ◌ 保证用户使用的 Web 浏览器和计算平台上的软件不会被病毒感染或被恶意程序破坏。
    - ◌ 确保用户的隐私和私人信息不会遭到窃取和破坏。
  - ✧ 保护连入因特网设备的安全
    - ◌ 保护路由器、交换机等设备的正常运行。
    - ◌ 保证不被黑客安装监控以及后门程序。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 安全技术主要包括三大类：
  - ✧ Web 服务器安全技术 - Web Server Security
  - ✧ Web 应用安全技术 - Web Application (Services) Security
  - ✧ Web 浏览器安全技术 - Web Browser Security

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 服务器安全技术
  - ✧ 安全配置 Web 服务器
    - ◌ 充分利用 Web 服务器本身拥有的安全机制 (如主目录权限设定、用户访问控制、IP 地址许可等) 对服务器进行合理有效的配置，确保 Web 服务的访问安全。
  - ✧ 网页防篡改技术
    - ◌ 将网页监控与恢复结合在一起，通过对网站的页面进行实时监控，主动发现网页页面内容是否被非法改动，提供立即恢复被篡改的网页页面的机制。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 服务器安全技术
  - ✧ 反向代理技术
    - ◌ 当外网用户访问网站时，采用代理与缓存技术，使得被访问的是反向代理系统，而不是真正的 Web 服务器系统，从而规避外网用户对 Web 服务器的攻击。反向代理系统会分析用户的请求，以确定是直接从本地缓存中提取结果，还是把请求转发到 Web 服务器。由于代理服务器上不需要处理复杂的业务逻辑，代理服务器本身被入侵的机会极低。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 服务器安全技术
  - ✧ 蜜罐技术
    - ◌ 蜜罐系统模拟 Web 服务器的行为，判别外来访问是否对应用服务器及后台数据库系统有害，从而有效防范各种已知及未知的攻击行为。
    - ◌ 对于通常的网站或邮件服务器，攻击流量会被合法流量所淹没。蜜罐进出的数据大部分是攻击流量，因而易于分析数据、查明攻击者行为。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

- **Web Security Technology**
  - Web 服务器安全技术
    - 数字水印嵌入和水印检测技术
      - 通过数字证书、加密技术和数字签名技术可以保证原始数据和水印信息在因特网上的安全传输。基于数字水印的日志文件技术能够确保日志的完整性、连续性和不可抵赖性，为日后日志审计、入侵取证提供权威依据。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 服务器安全技术
  - ✧ 身份认证技术
    - ◌ 在普通的 Web 应用中，攻击者可以假冒合法的用户或者服务器进行操作以达到非法目的。
    - ◌ 身份认证技术用于防止非法读取信息或执行非法操作，目前主要有三种形式：简单身份认证 (帐号/口令)、强度身份认证 (公钥/私钥)、基于生物特征的身份认证。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 应用安全技术
  - ✧ 访问控制技术
    - ◌ 访问控制通过技术手段准许或者限制对象的访问能力和范围，从而限制其对关键资源和敏感数据的访问，阻止非法用户入侵或合法用户误操作所导致的破坏。访问控制是保证信息机密性和完整性的关键技术。
    - ◌ 典型的访问控制技术有自主访问控制、强制访问控制、基于角色的访问控制和基于属性的访问控制等。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 应用安全技术
  - ✧ 数据保护技术
    - ◌ 数据保护主要由数据加密技术实现，例如
      - 对称加密算法
      - MD5 (SHA)、AES 和 RSA 混合加密算法
      - 3-DES 加密算法等。
  - ✧ 安全代码技术
    - ◌ 在应用服务代码编写过程中引入安全编程的思想，使得编写的代码免受隐藏字段、溢出、参数篡改等攻击。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 浏览器安全技术
  - ✧ Web 浏览器应用程序的基本功能是把图形用户界面 (GUI) 请求转换为 HTTP 请求传送给应用服务器，并把服务器的 HTTP 响应转换为 GUI 显示内容返回给用户。
  - ✧ 浏览器升级
    - ◌ 关注最新的补丁程序。
  - ✧ Java 安全限制
    - ◌ Java 在最初设计时便考虑了一些安全措施，如安全沙盒模型 (security sand box model)、签名小应用程序代码限制和细粒度访问控制等都可以用于限制哪些安全敏感资源可被访问，以及如何被访问。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- Web 浏览器安全技术
  - ✧ SSL 加密
    - ◌ SSL 可内置于 Web 浏览器，从而保证在 Web 浏览器和服务器之间的安全传输。在 SSL 握手阶段，服务器端的证书可被发送给 Web 浏览器，用于认证特定服务器的身份。同时，客户端的证书可被发送给 Web 服务器，用于认证特定用户的身份。
  - ✧ 匿名站点
    - ◌ 出于隐私保护的考虑，一些 Web 站点在 HTTP 请求中删除用户隐私相关信息，再把用户的 Web 请求重定向到其他 Web 站点。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Security Technology

- 云安全技术
  - ✧ 云安全概念紧随云计算、云存储之后出现，其中融合了网格计算、未知病毒行为判断等新兴技术。
  - ✧ 云安全技术可以依靠庞大的网络服务能力，对威胁信息进行实时采集、分析以及处理，把对计算机和网络的攻击扼杀在到达之前。
  - ✧ 云安全技术可以提高安全防护的主动性，扩大防护面，及时处理更多种类的安全威胁。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- Web 应用的安全漏洞主要有：
  - ✧ 已知弱点和错误配置
  - ✧ 隐藏字段
  - ✧ 后门和调试漏洞
  - ✧ XSS 跨站脚本攻击
  - ✧ 参数篡改
  - ✧ 更改 cookie
  - ✧ 输入信息控制
  - ✧ 缓冲区溢出
  - ✧ 直接访问浏览

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- 已知弱点和错误配置
  - ✧ 已知弱点包括 Web 应用所使用的操作系统和第三方应用程序中的错误或者可以被利用的漏洞。这个问题也涉及配置错误、包含不安全的默认设置或管理员没有进行安全配置的应用程序。
    - ◌ 例：Web 服务器被配置成可以让任何用户从系统上的任何目录路径通过，从而导致存储在 Web 服务器上的一些敏感信息的泄露，如口令、源代码或客户信息等。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- 隐藏字段
  - ✧ 在许多 web 应用中，隐藏的 html 格式字段被用来保存系统口令或商品价格。实际上这些字段并不很隐蔽，它们的内容可以轻易被类似“查看源代码”的命令所截获。同时许多 Web 应用并没有拒绝用户恶意修改 html 源文件中的这些字段。大多数 web 应用程序没有对返回网页进行验证，它们默认为输入数据和输出数据是一致的，这就给隐藏字段攻击提供了机会。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- 后门和调试漏洞
  - ✧ 开发人员常常建立一些程序后门以方便调试排除应用程序的故障。这些程序后门经常驻留在一些已经交付使用的 web 应用系统中，形成安全漏洞。一些常见的程序后门使用户不用口令就可以登录或者访问允许直接进行应用配置的特殊的 URL。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- XSS 跨站脚本攻击
  - ✧ XSS (Cross Site Script) 跨站脚本攻击指的是攻击者在 Web 页面插入恶意 html 代码，当用户浏览该页面时，嵌入页面的 html 代码会被执行，从而达到攻击者的特殊目的。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- 参数篡改
  - ✧ 参数篡改包括操纵 URL 字符串检索用户的以其他方式得不到的信息。访问 Web 应用的后端数据库是通过常常包含在 URL 中的 SQL 调用来进行的。恶意的用户可以操纵 SQL 代码，以便将来有可能检索一份包含所有用户、口令、信用卡号的清单或者储存在数据库中的任何其他数据。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- 更改 cookie
  - ✧ 更改 cookie 指攻击者修改存储在 cookie 中的数据。网站常常将一些包括用户 ID、口令、帐号等的 cookie 存储到用户系统上。通过改变这些值，恶意用户可以访问不属于他们的帐户。攻击者也可以窃取用户的cookie 并访问用户的帐户，而不必输入 ID 和口令或进行其他验证。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- 输入信息控制
  - ✧ 输入信息控制主要指攻击者通过控制由 CGI (Common Gateway Interface 通用网关接口) 脚本处理的 html 格式中的输入信息来运行系统命令。例如，使用 CGI 脚本向另一个用户发送信息的形式可以被攻击者控制，用以将服务器的口令文件邮寄给恶意的用户或者删除系统上的所有文件。

8.3 Web Security Primer

8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- 缓冲区溢出
  - ✧ 恶意用户向 web 服务器发送数据造成缓冲区溢出，缓冲区溢出的部分数据可能会覆盖系统栈的内容，达到执行非法代码或者瘫痪系统的目的。Web 应用缓冲区溢出攻击的典型例子也涉及到 html 文件。如果 html 文件上的一个字段中的数据足够大的话，它就能制造一个缓冲器溢出条件。

8.3 Web Security Primer

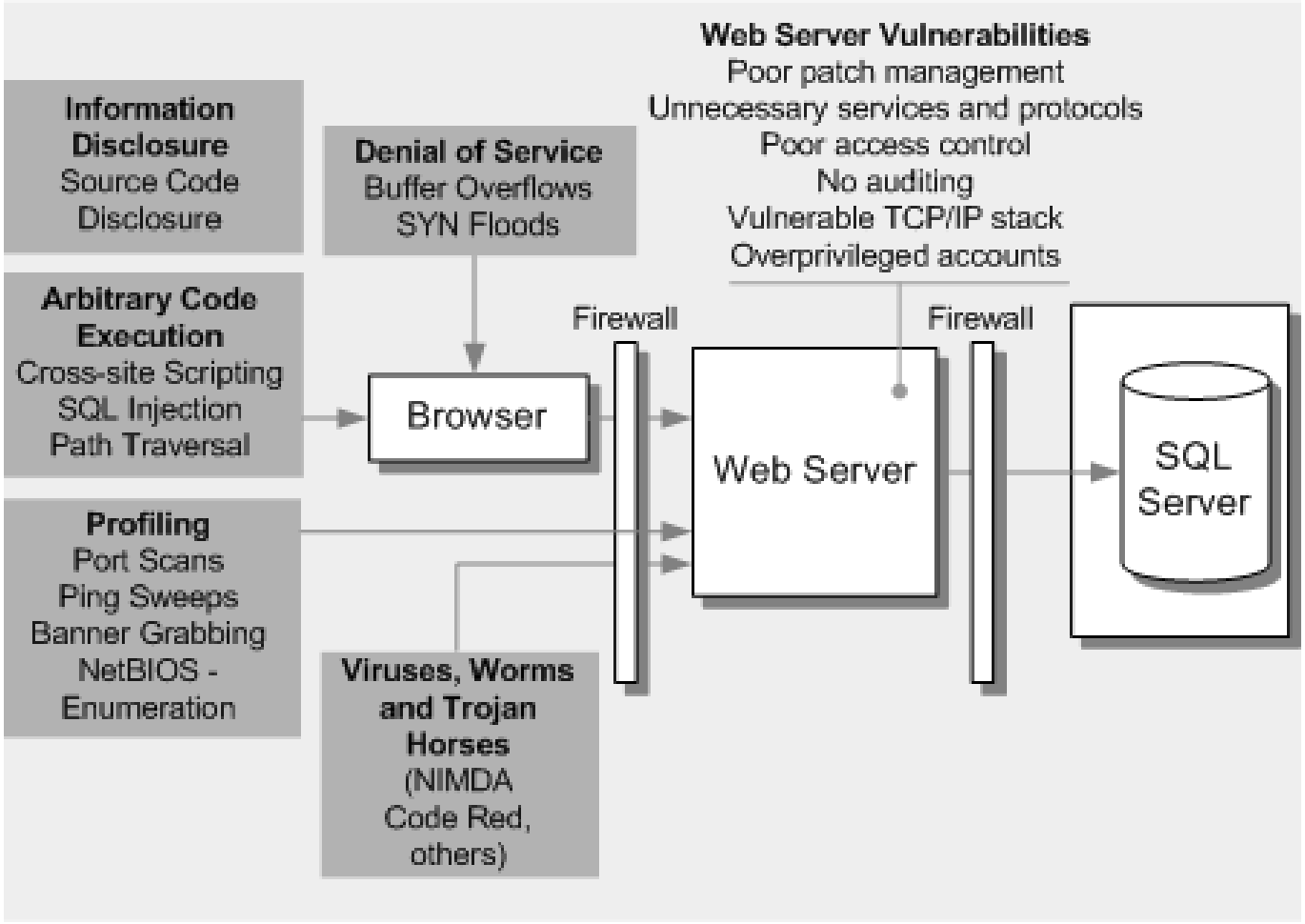
8.3.1 Web Security - Beginning

❑ Web Application Security Flaws

- 直接访问浏览
  - 直接访问浏览指直接访问应该需要验证的网页。没有正确配置的 Web 应用程序可以让恶意的用户直接访问包括有敏感信息的 URL 或者使提供收费网页服务的公司丧失收入。
- 参考：
  - ✧ OWASP Top 10 – 2017: The Ten Most Critical Web Application Security Risks

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities



8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

- **Threats to Web Server**
  - 分析
  - 拒绝服务
  - 未授权访问
  - 任意代码执行
  - 特权提升
  - 恶意代码植入
    - ✧ 病毒、蠕虫、木马

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

❑ Threats to Web Server

- 分析 (或称主机枚举)
  - ✧ 分析是用来收集 Web 站点信息的探索性过程。攻击者可以利用这些信息攻击站点已知的弱点。
  - ✧ 使服务器易受分析影响的常见漏洞包括：
    - ◌ 不必要的协议
    - ◌ 打开的端口
    - ◌ Web 服务器在旗标中提供的配置信息
  - ✧ 常见的用于分析的攻击包括：
    - ◌ 端口扫描
    - ◌ 标志获取 (识别 Web 服务器类型和版本)
    - ◌ Ping 扫视
    - ◌ NetBIOS 和服务器消息块 (SMB) 枚举

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

- **Threats to Web Server**
  - 分析 (续)
    - 对策：
      - 阻塞所有不必要的端口
      - 阻塞 ICMP (Internet Control Message Protocol) 流量
      - 禁用不必要的协议 (例如 NetBIOS 和 SMB)

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

❑ Threats to Web Server

- 拒绝服务
  - ✧ 拒绝服务攻击使得 Web 服务器被恶意的服务请求淹没，从而无法对合法客户端的请求做出响应。
  - ✧ 增加拒绝服务攻击可能性的漏洞包括：
    - ◌ 脆弱的 TCP/IP 堆栈配置
    - ◌ 未安装修补程序的服务器
  - ✧ 常见的拒绝服务攻击包括：
    - ◌ 网络级 SYN 洪泛；分布式请求洪泛攻击
    - ◌ 缓冲区溢出
  - ✧ 对策：
    - ◌ 加固 TCP/IP 堆栈
    - ◌ 及时对系统软件应用最新的补丁程序和更新。

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

❑ Threats to Web Server

- 未授权访问
  - ✧ 没有正当授权的用户获取了访问受限信息或者执行受限操作所需的权限。
  - ✧ 导致未授权访问的常见漏洞包括：
    - ◌ 脆弱的 IIS Web 访问控制，包括 Web 权限
    - ◌ 脆弱的 NTFS 权限
  - ✧ 对策：
    - ◌ 使用安全的 Web 权限、NTFS 权限
    - ◌ 使用 .NET Framework 访问控制机制 (包括 URL 授权)

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

❑ Threats to Web Server

- 任意代码执行
  - ✧ 攻击者在服务器上运行恶意代码以威胁服务器资源的安全或者对下游系统实施其他攻击。
  - ✧ 可能导致恶意代码执行的常见漏洞包括：
    - ◌ 脆弱的 IIS 配置
    - ◌ 未安装修补程序的服务器
  - ✧ 常见的代码执行攻击包括：
    - ◌ 路径遍历
    - ◌ 导致代码注入的缓冲区溢出
  - ✧ 对策：
    - ◌ 将 IIS 配置为拒绝带有“../”的 URL 以防止路径遍历
    - ◌ 用限制性访问控制列表 (ACL) 锁定系统命令和实用工具
    - ◌ 及时安装新的补丁程序和更新。

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

❑ Threats to Web Server

- 特权提升
  - ✧ 攻击者使用特权进程帐号运行恶意代码。
  - ✧ 使 Web 服务器容易遭受特权提升攻击的常见漏洞包括：
    - ◌ 特权过高的进程帐号
    - ◌ 特权过高的服务帐号
  - ✧ 对策：
    - ◌ 使用最低特权帐号以及使用最低特权服务和用户帐号运行进程。

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

❑ Threats to Web Server

- 恶意代码植入
  - ✧ 恶意代码有几种变种，包括：
    - ◌ 病毒：执行恶意操作并导致操作系统或者应用程序崩溃的程序。
    - ◌ 蠕虫：可以自我复制和自我持续的程序。
    - ◌ 木马：隐藏的恶意程序。
  - ✧ 在许多情况下，恶意代码直至开始消耗系统资源并减慢或者阻碍了其他程序的执行时，才会被注意到。
  - ✧ 导致容易遭受恶意代码攻击的常见漏洞包括：
    - ◌ 未安装修补程序的服务器
    - ◌ 运行不必要的服务
    - ◌ 不必要的 ISAPI 筛选器和扩展

8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

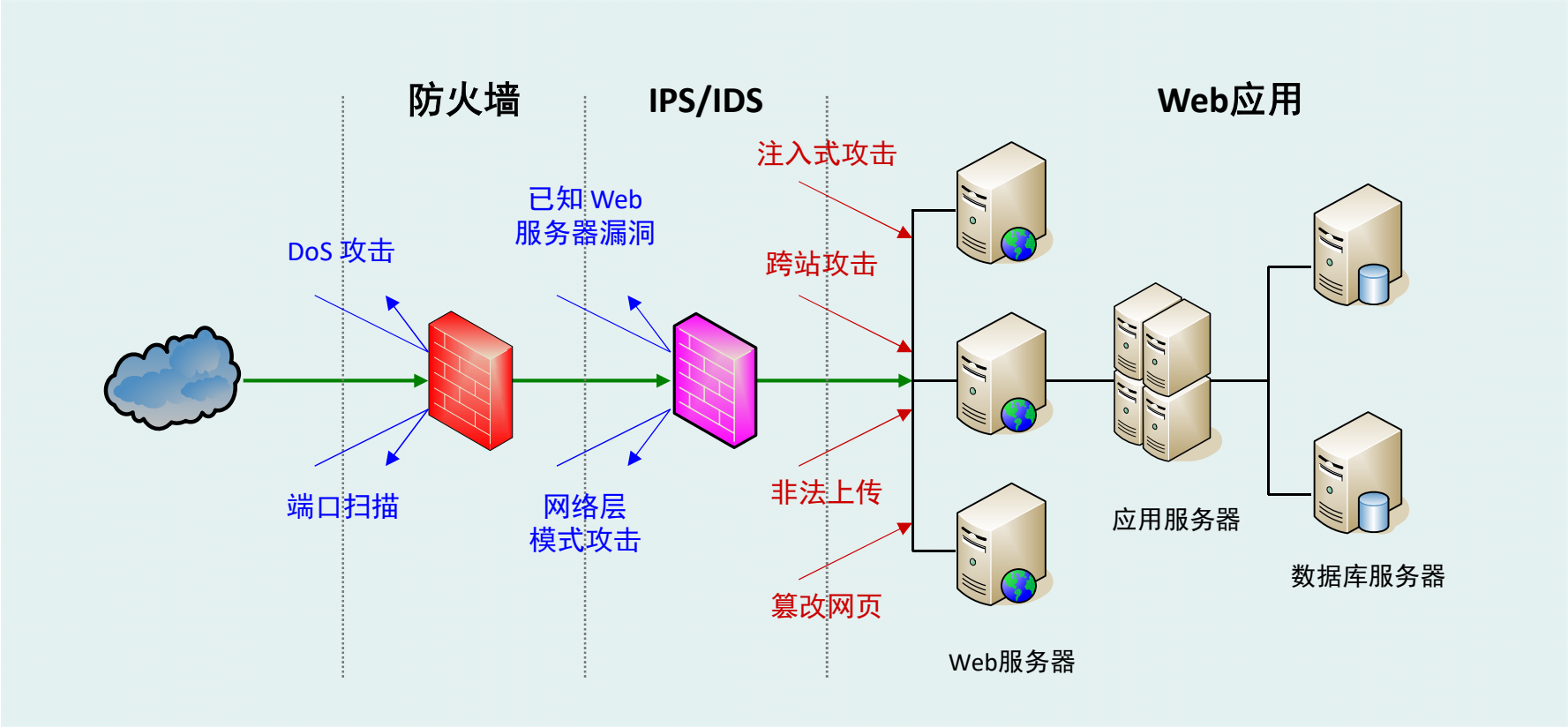
❑ Threats to Web Server

- 恶意代码植入 (续)
  - ✧ 对策：
    - ◌ 及时应用软件补丁程序
    - ◌ 禁用未用的功能 (例如未用的 ISAPI 筛选器和扩展)
    - ◌ 用最低特权帐号运行进程以减小出现攻击时破坏的范围。
      - ISAPI: Internet Server Application Programming Interface, used for IIS.
      - Different from CGI

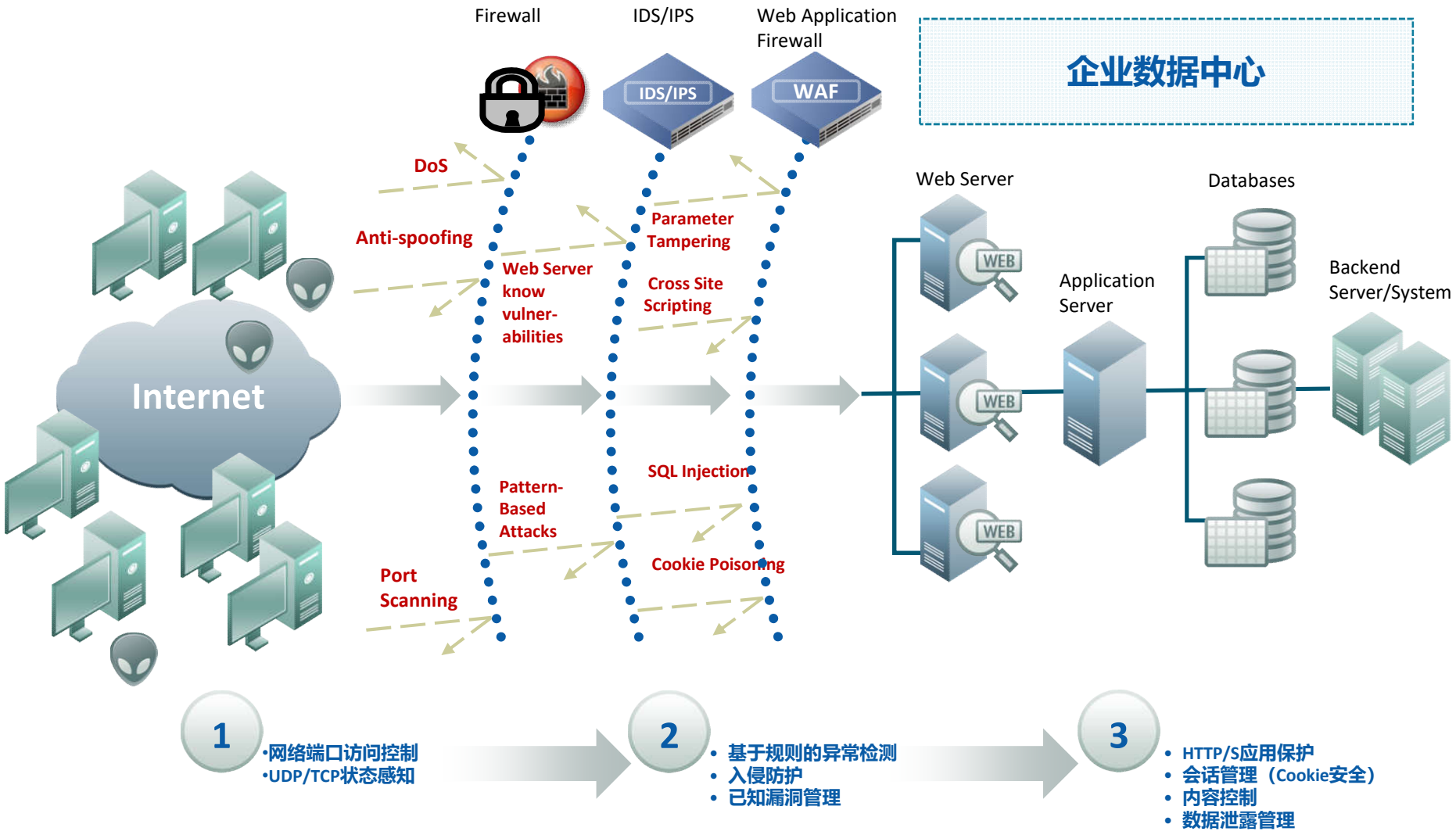
8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

❑ Overview of Web Server Security



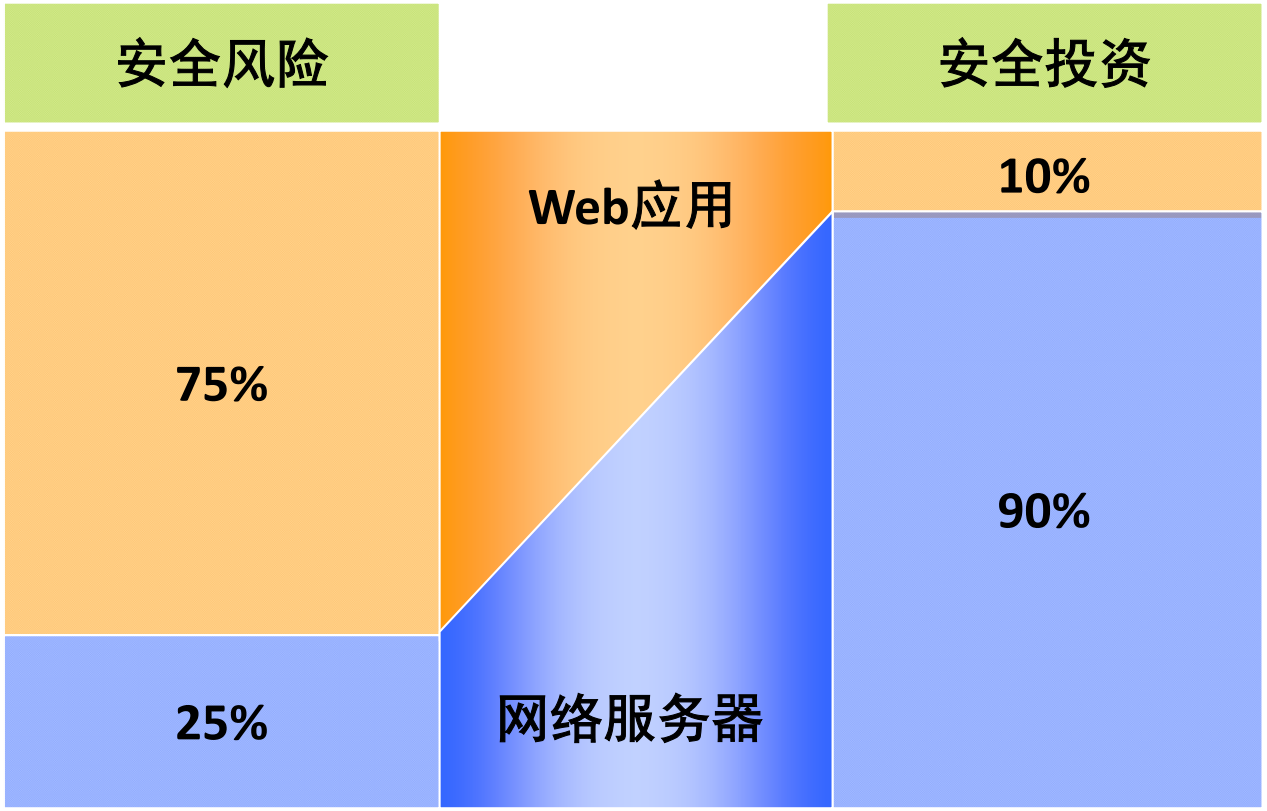
8.3 Web Security Primer



8.3 Web Security Primer

8.3.2 Web Server Vulnerabilities

- **Web Security Risk and Investment**



8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

- What is WS-Security
  - ✧ WS-Security is a flexible and feature-rich extension to SOAP to apply security to web services. It is a member of the WS-* family of web service specifications and was published by OASIS (Organization for the Advancement of Structured Information Standards).
- WS-Security Mechanisms
  - ✧ WS-Security describes three main mechanisms:
    - ◌ How to sign SOAP messages to assure integrity. Signed messages provide also non-repudiation.
    - ◌ How to encrypt SOAP messages to assure confidentiality.
    - ◌ How to attach security tokens to ascertain the sender‘s identity. (如何附加安全令牌以确认发送者的身份)

8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

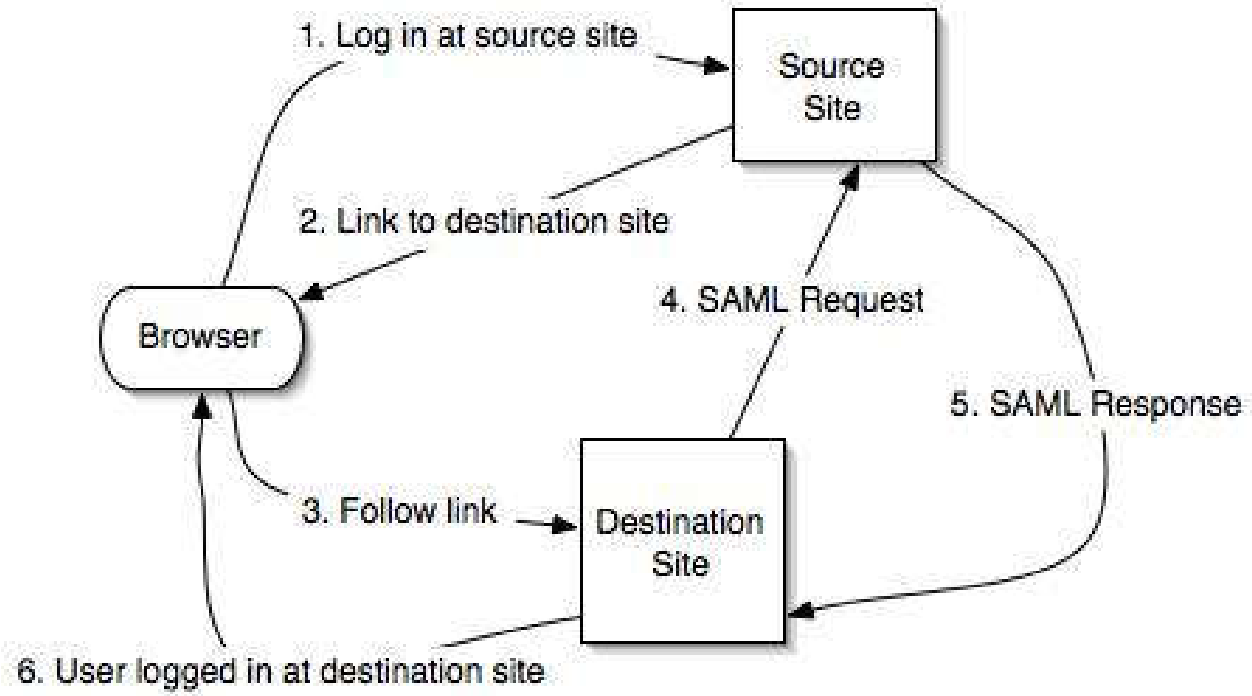
- WS-Security Mechanisms
  - ✧ The specification allows a variety of signature formats, encryptions algorithms and multiple trust domains, and is open to various security token models, such as:
    - ◌ X.509 certificates
    - ◌ Kerberos tickets
    - ◌ User ID/Password credentials
    - ◌ SAML-Assertion (SAML, Security Assertion Markup Language)
    - ◌ Custom defined token

8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

- WS-Security Mechanisms
  - ✧ A scenario of applying SAML-Assertion



8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

- Web 服务安全模型
  - ✧ 为了解决 Web 服务安全通信问题，微软和 IBM 共同定义了一个 Web 服务安全模型。该模型以 WS- Security 规范为核心，保证了 SOAP 消息的安全性。通过定义 WS- Policy、WSTrust 等规范，保证了上层应用系统的安全性
  - ✧ Web 服务安全性模型引入了一个由各个相互联系的规范组成的集合，这些规范描述了把安全性功能程序放到 Web 服务环境中的方法。体系结构被设计成允许对规范进行混合匹配，使实现者能够按需部署。这些规范中的第一个文档 "Web Security" (Web 服务安全性) 提供了把消息完整性和机密性功能程序添加到 Web 服务中所必需的基本元素，并且提供把安全性令牌 (例如，数字证书和 Kerberos 票据) 关联到 SOAP 消息的方法。

8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

- WS- Security 规范
  - ✧ WS- Security 规范提出了一套标准的 SOAP 安全扩展方法，通过对消息的加密、数字签名以及认证，保证了消息端对端的安全传输。
    - ① 身份验证：通过 SOAP Header 传递用于验证用户身份的安全令牌 (SecurityToken) 。
    - ② 消息加密：使用 XML Encryption 技术保证所传输消息的机密性。
    - ③ 消息数字签名：使用 XML Signature 技术 (W3C Rec.) 保证消息的完整性和一致性。

8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

- WS- Policy 和 WS- SecurityPolicy 规范
  - ✧ WS- Policy 定义了 Web 服务策略模型和策略引用包含机制。策略由策略表达式表示。其他 Web 服务规范可以使用 WS- Policy 提供的框架来定义相应的策略机制用以描述相应的服务需求、偏好和能力。

8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

- WS- Policy 和 WS- SecurityPolicy 规范
  - ✧ WS- SecurityPolicy 以 WS- Policy 规范为基础，定义了可用于指定安全策略的 XML 元素，由元素明确指定安全策略。换言之，WS- SecurityPolicy 指定了如何使用 WS- Policy 语言表达 WS- Security 安全策略。这些策略阐明了针对特殊情况的特定安全需求，比如采用何种验证手段、客户端接受哪种数字签名算法、服务器端支持哪些加密算法等。WS- SecurityPolicy 定义的安全策略 XML 元素包括：
    - ◌ <SecurityToken> 指定安全令牌类型
    - ◌ <Integrity> 指定数字签名的手段
    - ◌ <Confidentiality> 指定加密算法
    - ◌ <Visibility> 指定需解密的消息片断

8.3 Web Security Primer

8.3.3 Web Services Secure Model

- **WS-Security**
  - 其它协议
    - WS- Trust：该规范描述了如何在 Web 服务环境中建立企业之间的信任关系。
    - WS- Privacy：该规范描述了如何把隐私权策略以及首选项与 Web 服务相关联。
    - WS- SecureConversation：该规范描述了如何将集合消息作为更复杂的企业事务的一部分实现安全交换。

8.3 Web Security Primer

8.3.3 Web Services Secure Model

- **WS-Security**
  - 其它协议
    - WS- Federation：该规范描述了一个模型，用于把不兼容的安全性机制，或将部署在不同域中的、类似的机制进行集成。
      - 例如，如果两个 IBM 业务伙伴都实现了基于 PKI 的身份认证基础架构，或者其中一个伙伴实现了 Kerberos 系统，WS-Federation 规范将提供一个关于如何应用 Web 服务技术把这些体系联系到一起的指南。
    - WS- Authorization：该规范描述了如何在 Web 服务基础架构中提供应用程序授权请求和决定。

8.3 Web Security Primer

8.3.3 Web Services Secure Model

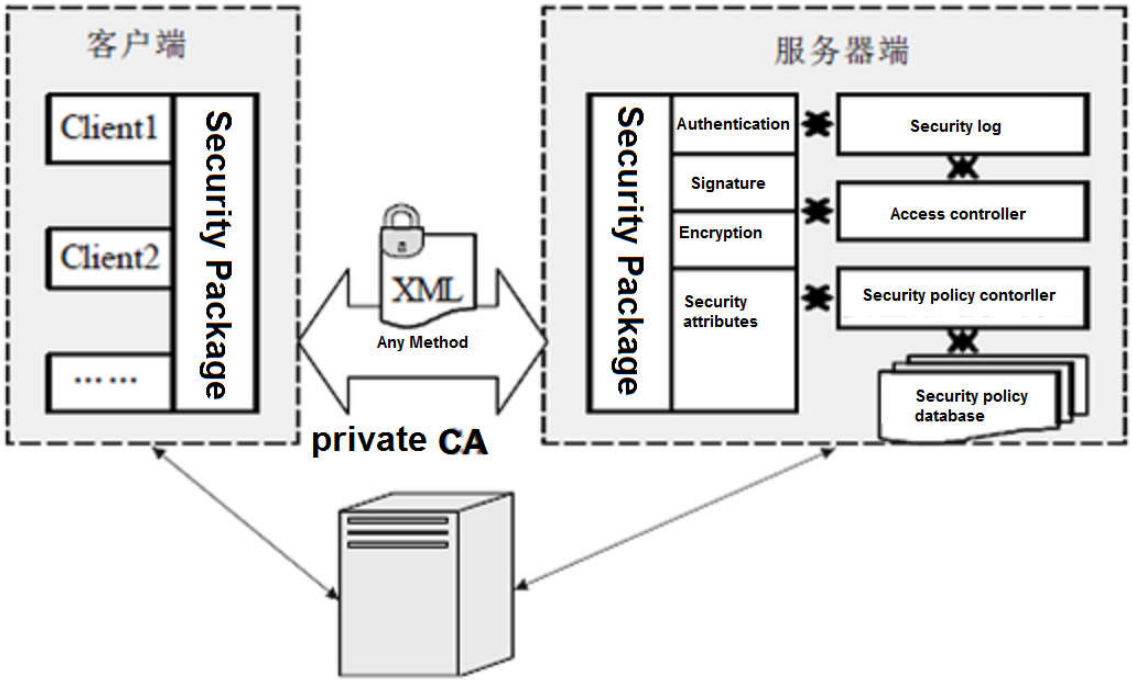
- **WS-Security**
  - See also:
    - http://www.microsoft.com/china/technet/security/guidance/secmod10.mspx#E2D
    - http://msdn.microsoft.com/webservices/

8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

- See also:
  - ✧ http://msdn.microsoft.com/zh-cn/library/ff648653.aspx#E4AA



8.3 Web Security Primer

8.3.3 Web Services Secure Model

❑ WS-Security

- See also:
  - ✧ 该系统基于 SOAP 头的扩展，分别实现 SOAP 加密、SOAP 签名、SOAP 认证与授权、SOAP 安全属性扩展 (时间戳等) 。对于访问控制, 设计了 Web 服务访问控制器，该控制器的实现主要基于 X- RBAC 模型，动态控制用户对 Web 服务的访问权限。系统还包括一个私有 CA 中心，主要负责密钥和证书的生成和管理。此外，安全日志保证了模型系统具备审计的功能。安全策略管理器则负责对安全策略文件的管理，安全策略库保存着 Web 服务所对应的特定安全策略文件。

8.3 Web Security Primer

8.3.4 Prevention of Malicious Code

❑ Malicious Code

- Malicious Code consists of programming (code, scripts, active content, and other software running in browsers) designed to disrupt or deny operation, gather information that leads to loss of privacy or exploitation, gain unauthorized access to system resources, and other abusive behavior.
  (1) It's malicious
  (2) It's a program
  (3) It runs in browsers
  (4) It's embedded in a HTML document usually
  (5) It utilizes a bug or a hole usually

8.3 Web Security Primer

8.3.4 Prevention of Malicious Code

❑ Principle & Prevention

- ActiveX (Microsoft)
  - ✧ Downloaded automatically through <OBJECT> by IE
  - ✧ Follow Authenticode
    - ◌ Nonfeasance
    - ◌ safe-for-script
      - Scriptlet
      - Eyedog

8.3 Web Security Primer

8.3.4 Prevention of Malicious Code

❑ Principle & Prevention

- JAVA/Applet (SUN)
  - ✧ Use sandbox environment, perfect theoretically but too big to achieve
  - ✧ JVM (Java Virtual Machine) bugs
- JavaScript & Active Scripting
  - ✧ Easy-to-use and functional
  - ✧ Browser bugs

8.3 Web Security Primer

8.3.5 SSL/Https

- **SSL**
  - Application of SSL in Web Application (https)
- **HTTPS** (Hypertext Transfer Protocol Secure)
  - What is HTTPS
    - A combination of the Hypertext Transfer Protocol (HTTP) with SSL/TLS protocol to provide encrypted communication and secure identification of a network web server.
  - Processes
    - A web site get its certificate from CA (VeriSign, Microsoft)
    - Browsers check this certificate
    - Web server and browser communicate with each other, following HTTP protocol which works on SSL

8.3 Web Security Primer

8.3.5 SSL/Https

❑ HTTPS

- Differences from HTTP
  - ✧ URL: http:// and https://
  - ✧ Port: 80 and 443
  - ✧ Security
- Network layers
  - ✧ Application layer: HTTP+SSL/TSL
- Server setup
  - ✧ Certificate
- Acquiring certificates
  - ✧ Not free
- Use as access control
- In case of compromised private key
  - ✧ OCSP (Online Certificate Status Protocol)

References

1. *Joel Scambray*, Hacking Exposed Web Applications, McGraw-Hill, 2011
2. 中国信息安全测评中心, Web 系统安全和渗透性测试基础, 航空工业出版社 2009
3. *Mike Andrews*, How to Break Web Software, Pearson, 2006
4. http://msdn.microsoft.com/zh-cn/library/ff648653.aspx#E4AA
5. 韩涛,Web 服务安全模型研究与实现,计算机工程.2006.5
6. http://www.microsoft.com/china/technet/security/guidance/secmod10.mspx#E2D
7. http://www.oasis-open.org
8. http://msdn.microsoft.com/zh-cn/library/ff648653.aspx#E4AA
9. http://www.microsoft.com/china/technet/security/guidance/secmod10.mspx#E2D
10. http://msdn.microsoft.com/webservices/
11. CERT-CC: http://www.cert.org.cn/

End of Chapter 8