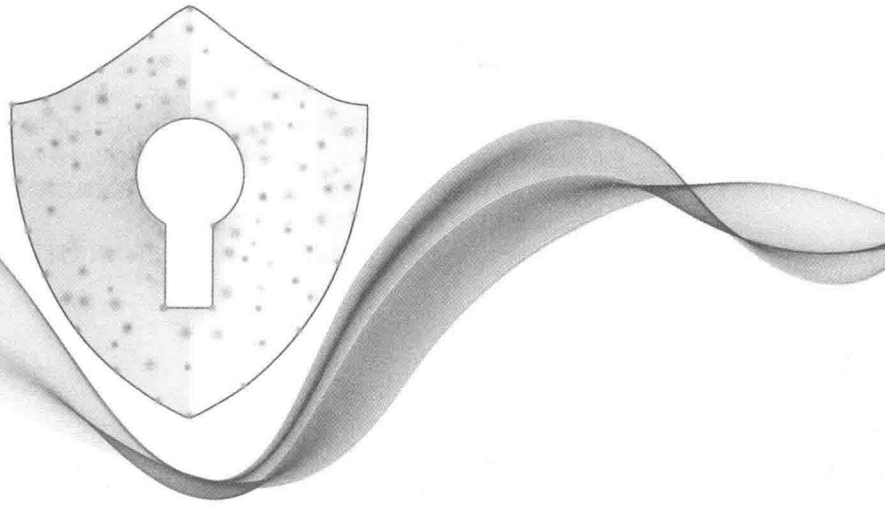
Web系统攻防技术与实践

国网湖南省电力公司电力科学研究院　组编

漆文辉　主编

中国电力出版社
CHINA ELECTRIC POWER PRESS



Web系统攻防技术与实践

国网湖南省电力公司电力科学研究院　组编

漆文辉　主编

中国电力出版社
CHINA ELECTRIC POWER PRESS

内 容 提 要

本书从攻击和防护两个角度系统地讲述了 Web 安全的相关理论和案例。其中，攻击技术选取了 OWASP TOP 10 中最常见也是危害最大的几类技术进行介绍，包括跨站脚本 XSS、跨站请求伪造 CSRF、SQL 注入、文件上传漏洞和文件包含漏洞等，每一类技术除了介绍其基本概念和技术原理外，还通过真实发生的实际案例进行深入分析。防护技术方面，在介绍攻击技术的每一章最后，都会有针对性地介绍此类攻击手段的最有效防护方法和原理。

此外，本书还通过两个独立的章节，系统地介绍了 Web 日志和主机日志分析的方法，介绍如何通过分析日志文件来发现入侵行为，进而针对安全威胁采取对应的防范措施。

本书实例丰富、典型，实战性强，非常适合电力系统信息安全人员使用。

图书在版编目（CIP）数据

Web 系统攻防技术与实践 / 漆文辉主编；国网湖南省电力公司电力科学研究院组编. —北京：中国电力出版社，2017.12（2018.5重印）

ISBN 978-7-5198-1124-2

Ⅰ. ①W… Ⅱ. ①漆…②国… Ⅲ. ①互联网络–安全技术 Ⅳ. ①TP393.408

中国版本图书馆 CIP 数据核字（2017）第 219457 号

出版发行：中国电力出版社
地　　址：北京市东城区北京站西街 19 号（邮政编码 100005）
网　　址：http://www.cepp.sgcc.com.cn
责任编辑：袁　娟（010–63412561）　夏华香 huaxiang-xia@sgcc.com.cn
责任校对：常燕昆
装帧设计：郝晓燕　左　铭
责任印制：邹树群

印　　刷：三河市百盛印装有限公司
版　　次：2017 年 12 月第一版
印　　次：2018 年 5 月北京第二次印刷
开　　本：787 毫米×1092 毫米　16 开本
印　　张：16.5
字　　数：374 千字
印　　数：1001—2000册
定　　价：68.00 元

前 言

近年来，随着互联网技术的发展，以及近期“互联网+”概念的提出，越来越多传统行业融合互联网发展新形态、新业态，如今许多企业的业务开展都离不开 Web 应用系统。Web 应用系统在提供便捷性的同时，也带来了不小的安全隐患。正是由于其开放性和访问的便捷性，Web 应用一直是黑客的重点攻击对象，据 Gartner 的数据表明，当前网络上 75%以上的攻击都是针对 Web 应用进行的。

Web 安全的攻与防是密不可分的。只有通过从用户或入侵者的角度对目标系统进行渗透测试，了解其攻击的手段和原理，才能更加有的放矢的采取防护措施，取长补短，实现最有效的防御。本书也是沿用这种思路，试图站在入侵者的角度，研究 Web 应用系统攻击与防护的关键技术，并通过搭建实际漏洞环境和对大量实际案例讲解与分析，让读者对 Web 系统攻防技术有更直观的体会和更深入的认识。

目 录

第一章

Web系统安全概述

随着互联网的普及与发展，Web系统在电子商务、医疗保健、电子政务和企事业数据平台等领域得到广泛应用。伴随Web技术给商业活动带来便利的同时，针对Web系统的攻击越来越多，Web系统安全已成为一个日益重要的问题。权威机构分析报告指出，Web系统漏洞数量占已发现漏洞总数的50%以上。美国互联网安全厂商CENZIC的分析报告显示，2009年网络攻击事件有82%都是针对Web系统发起的。另据360互联网安全中心调查显示，2013年国内平均每天有3500多家网站遭到35万次的各类攻击；65.5%的网站存在安全漏洞，8.7%的网站遭到篡改，33.7%的网站被植入了后门。

作为一个信息安全人员首先应该了解Web系统的架构，理解其具有脆弱性根源；其次，应了解当今Web系统面临的主要威胁。本章将在Web系统架构上，介绍Web系统安全防护技术。

第一节 Web系统基础

一、Web系统架构

Web（World Wide Web，WWW）是互联网的总称，包括互联网的使用环境、工具、氛围、内容等。Web系统（Web Applications）是一种运行于Web和企业内部TCP/IP互联网络之上的浏览器/服务器（Browser/Server）架构的应用程序。如图1-1所示，一个完整的Web系统通常包含Web服务器、中间件服务器、数据库服务器和客户端等多个系统环节。在客户端用户在浏览器上发送HTTP协议请求，经过互联网和Web服务器交互，并将请求转化为对数据库服务器的查询或更新，最后将结果以页面的形式返回到用户浏览器中。

二、Web系统工作原理

用户在客户端的浏览器地址栏上输入统一资源定位符（Uniform Resource Locators，URL），浏览器首先请求域名服务（Domain Name Service，DNS），通过DNS获取相应域名对应的IP，然后通过IP地址找到IP对应的Web服务器，要求建立与其对应的TCP（Transmission Control Protocol）连接。待链接建立完成后，浏览器发送完超文本传输协

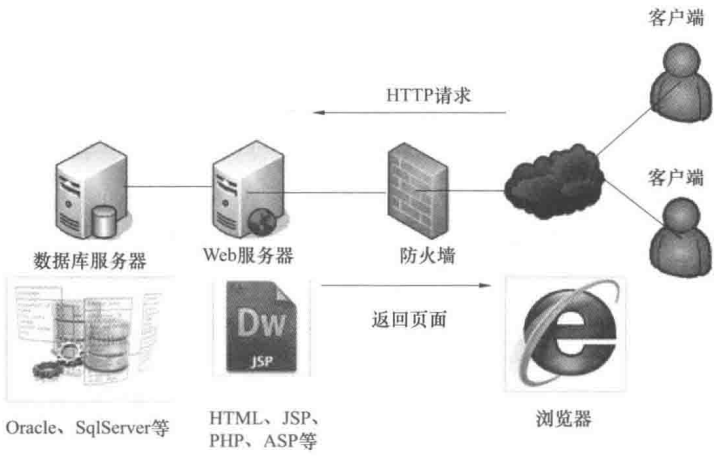


图 1-1　Web 系统结构

议（Hyper Text Transport Protocol，HTTP）请求包，请求 HTML（Hyper Text Markup Language）页面。服务器接收到请求包后才开始查找页面资源，服务器调用自身服务，返回 HTTP 响应包，客户端收到来自服务器的响应后开始渲染这个响应包里的 HTML 页面，等收到全部的内容后断开 Web 服务器之间的 TCP 连接。以浏览器访问 www.baidu.com 为例，Web 系统工作原理可归纳为以下六步，如图 1-2 所示。

（1）浏览器通过 DNS 获取相应域名 www.baidu.com 对应的 IP 为 111.13.100.91。

（2）用户在浏览器输入页面请求（URL），该请求从浏览器传送到 Web 服务器。

（3）Web 服务器收到请求后，查找并定位页面位置。

（4）Web 服务器创建 HTML 流。

（5）Web 服务器将 HTML 流通过网络传回到浏览器。

（6）浏览器处理 HTML，在客户端上渲染并显示该页面。

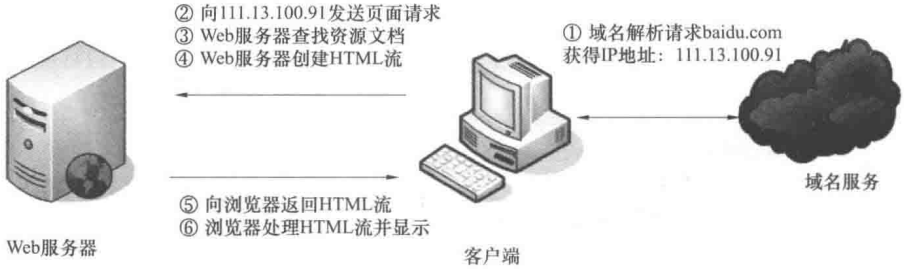


图 1-2　Web 系统工作原理

三、HTTP 协议基础

HTTP 用来传输网页、图像以及互联网上在浏览器与服务器间传输的其他类型文件。HTTP 是 Web 系统工作的核心协议，一般工作在 TCP 80 端口。

HTTP 由请求和响应两部分组成。当在浏览器中输入一个 URL 时，浏览器将创建并发送请求，该请求包含所输入的 URL 以及一些与浏览器本身相关的信息。当服务器收到

这个请求时将返回一个响应，该响应包括与该请求相关的信息以及位于指定 URL 的数据。直到浏览器解析该响应并显示出网页为止。

1. HTTP 请求

HTTP 请求的格式如下：

```
<request-line>
<headers>
<blank line>
[<request-body>]
```

在 HTTP 请求中，第一行必须是一个请求行，用来说明请求类型、要访问的资源以及使用的 HTTP 版本。紧接着是一个首部（header）小节，用来说明服务器要使用的附加信息。在首部之后是一个空行，再此之后可以添加任意的其他数据。

在 HTTP 中定义了多种请求类型，通常我们关心的只有 GET 请求和 POST 请求。只要在 Web 浏览器上输入一个 URL，浏览器就将基于该 URL 向服务器发送一个 GET 请求，以告诉服务器获取并返回什么资源。对于 www.baidu.com 的 GET 请求如下：

```
GET / HTTP/1.1
Host: www.baidu.com
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.36 (KHTML,like
Gecko) Chrome/32.0.1700.72 Safari/537.36
```

请求行的第一部分说明了该请求是 GET 请求，该行的第二部分是一个斜杠，用来说明请求的是该域名的根目录，该行的最后一部分说明使用的是 HTTP 1.1 版本。第 2 行是请求的第一个首部 HOST，指出请求的目的地。结合 HOST 和上一行中的斜杠，可以通知服务器请求的是 www.baidu.com。第三行首部 Connection，通常将浏览器操作设置为 keep-alive。最后一行中包含的是首部 User-Agent，服务器端和客户端脚本都能够访问它，它是浏览器类型检测逻辑的重要基础，该信息由使用的浏览器（在本例中是 Chrome）来定义，并且在每个请求中自动发送。注意，在最后一个首部之后有一个空行，即使不存在请求主体，这个空行也是必需的。

2. HTTP 响应

HTTP 响应的格式如下：

```
<status-line>
<headers>
<blank line>
[<response-body>]
```

在 HTTP 响应中，第一行必须是一个状态行，用来说明所请求的资源情况。紧接着是一个首部（header）小节，用来说明服务器要使用的附加信息。在首部之后是一个空行，再此之后可以添加任意的其他数据。对于 www.baidu.com 的响应如下：

```
HTTP/1.1 200 OK
Date:Tue,06 Jan 2015 07:30:04 GMT
ontent-Type:text/html; charset=utf-8
```

在本例中，状态行给出的 HTTP 状态代码是 200。状态行始终包含的是状态码和相应的简短消息，以避免混乱。最常用的状态码有：

（1）200（OK）：找到了该资源，并且一切正常。

（2）304（NOT MODIFIED）：该资源在上次请求之后没有任何修改。这通常用于浏览器的缓存机制。

（3）401（UNAUTHORIZED）：客户端无权访问该资源。这通常会使得浏览器要求用户输入用户名和密码，以登录到服务器。

（4）403（FORBIDDEN）：客户端未能获得授权。这通常是在 401 之后输入了不正确的用户名或密码。

（5）404（NOT FOUND）：在指定的位置不存在所申请的资源。

在状态行之后是一些首部。第一个首部 Date，用来说明响应生成的日期和时间。第二个首部 Content-Type，用于告诉客户端响应的数据类型，这样浏览器就根据返回数据的类型来进行不同的处理，如果是文本类型就直接显示内容，如果是 HTML 类型就用浏览器显示内容，如果是下载类型就弹出下载。本例中，首部 Content-Type 指定了 MIME 类型 HTML（text/html），其编码类型是 utf-8。

3. HTTP Cookie

HTTP 协议是一种无状态的协议，不能在服务器上保持一次会话的连续状态信息。HTTP 的无状态性不能满足某些应用的需求，给 Web 服务器和客户端的操作带来种种不便。在此背景下，提出 HTTP 的状态管理机制，即 HTTP Cookie（简称 Cookie）机制。Cookie 是 Web 服务器保存在用户浏览器中的一段文本文件，Web 服务器通过 HTTP 响应将它发送到用户浏览器中。当再次访问同一 Web 服务器时，浏览器将它原封不动地返回。Web 服务器之后可以利用这些信息来标识用户，实现对用户的短期跟踪。

HTTP Cookie 的运作机制如图 1-3 所示。当客户端浏览器首次请求访问 Web 服务器时，Web 服务器将 Cookie 信息写入 HTTP 响应中，并返回给客户端浏览器。客户端浏览器解析并保存 Cookie 信息，此后每次访问 Web 服务器，都会在 HTTP 请求数据中包含 Cookie 信息，服务器解析 HTTP 请求中的 Cookie，就能获得客户的相关信息。

Cookie 主要属性见表 1-1。

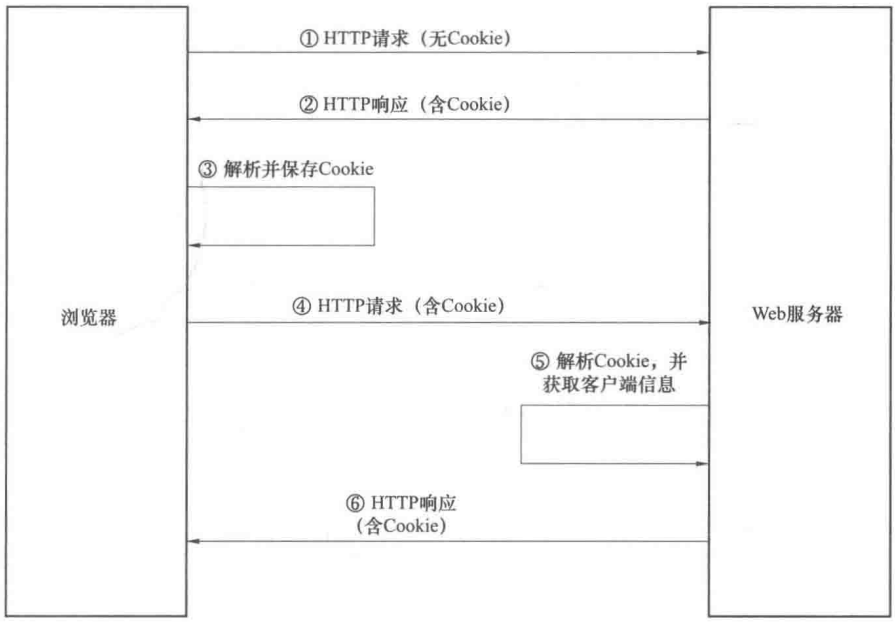


图 1-3　HTTP Cookie 运作机制

表 1-1　　Cookie 主 要 属 性

序号	属性名	定　义	注　释
1	Domain	设置当前 Cookie 所属域名	Domain 的值是默认创建 Cookie 的网页所在的服务器的主机名
2	Path	设置 Cookie 的所属路径	如果没有设置这个选项，Cookie 会和创建它的网页处于同一个目录下的网页
3	Expires	设置 Cookie 的有效期	如果没有设置这个选项，那么此 Cookie 有效期只是当前的 session，将在浏览器关闭时失效
4	secure	表示该 Cookie 只能用 HTTPS 传输	
5	httponly	表示此 Cookie 只能通过 HTTP 访问，不能被客户端脚本获取到	

Web 服务器创建 Cookie，并通过 HTTP 响应发送给客户端。在响应头 Set-Cookie 参数中，可设置 Cookie 的名称、值，以及各个属性，如下所示：

```
HTTP/1.1 200 OK
Set-Cookie:JSESSIONID=49187CF75167LK24E25.server1; Path=/procheck
Content-Type:text/html;charset=ISO-8859-1
Content-Length:377
Date:Mon,26 Jan 2015 07:40:07GMT
```

客户端浏览器通过 HTTP 请求发送 Cookie 时，不发送 Cookie 的各个属性，而只发送对应的名称和值，HTTP 请求如下：

```
Accept-Encoding: gzip, deflate
Host: localhost: 80
Connection: Keep-Alive
Cookie: JSESSIONID=49187CF75167LK24E25.server1
```

Cookie 可以维护客户端状态信息，是对 HTTP 协议的一种补充，以保持服务器和客户端的连续状态。Cookie 在给用户带来方便的同时，也导致了安全隐患。由于 Cookie 记录了用户 ID、密码等信息，其在浏览器和服务器之间传递，容易被攻击者拦截。攻击者不需要知道 Cookie 信息的含义，只需在有效期内重放，就可通过验证，冒充用户的身份访问服务器，给用户的利益带来损害。

4. HTTP Session

HTTP Session（Session）是 HTTP 协议中又一种在客户端与服务器之间保持状态的解决方案。Session 是一种保存上下文信息的机制，它为每一个用户创建一个 Session 变量。与 Cookie 机制不同，Session 值保存在服务器端，并通过 SessionID 查找对应的 Session。SessionID 以 Cookie、URL 重写或者隐藏表单变量的方式发送并保存在客户端浏览器中。如图 1-4 所示，以 Cookie 传递 SessionID 为例，服务器首先给每个 Session 分配一个唯一的 SessionID，并通过 HTTP 响应中的 SetCookie 将 SessionID 发送给客户端。当客户端发起新的请求时，再通过 HTTP 请求中的 Cookie 将 SessionID 回传到 Web 服务器，这样服务器能够找到这个客户端对应的 Session。

图 1-4　HTTP Session 运作机制

当 Web 服务器需要为某个客户端的请求创建一个 Session 时，服务器首先检查请求里是否已包含了一个 SessionID，如果已包含则说明以前已经为该客户端创建过 Session，服务器就根据 SessionID 把这个 Session 检索出来；如果客户端请求不包含 SessionID，则为此客户端创建一个 Session 并且生成对应的 SessionID。这个 SessionID 将通过此次响应返回给客户端。

Session 在下列情况下会被删除。

（1）Session 超时：在连续一定时间内，Web 服务器没有收到该 Session 所对应客户端的请求，并且这个时间超过了 Web 服务器设置的 Session 超时的最大时间。

（2）Web 应用程序调用 HttpSession.invalidate()方法。

（3）Web 服务器关闭或服务停止。

由于 Cookie 在传递过程中可能被攻击者窃取，攻击者就可获得保存在 Cookie 中的 SessionID，在其有效期内就可以凭此 SessionID 欺骗 Web 服务器，发动 Session 劫持攻击，登录 Web 服务器。

四、Web 系统漏洞

Web 系统的目的是为用户提供应用服务，即使有物理防护、软件防护、防火墙，仍然必须要允许一部分涉及正常服务的通信经过防火墙，Web 应用必需的 80 和 443 等端口是一定要开放的。可以顺利通过的这部分访问，可能是善意的，也可能是恶意的。恶意访问利用系统漏洞，执行各类恶意操作：偷窃、操控，或破坏 Web 应用数据，甚至利用 Web 系统作为攻击跳板，破坏企业的整个信息系统的可用性。2010 年，黑客利用微软 Word 办公软件漏洞，造成 Windows 版本 Word 出现栈溢出，通过下载恶意代码威胁用户系统数据安全；2011 年，Google Gmail 邮箱被入侵；2012 年 2 月，黑客对巴西银行进行分布式拒绝服务攻击等。2013 年 7 月 7 日晚间，韩国总统府、国防部、外交通商部等政府部门和主要银行、媒体网站等再次遭到分布式拒绝服务（Distributed Denial of Service，DDoS）攻击，瘫痪时间长达 4h。2014 年 4 月 8 日，互联网基础组件 OpenSSL 爆出“心脏出血”漏洞，此漏洞被认为是近年来危害最严重的安全漏洞，可以让黑客轻松在 HTTPS 开头网址服务器上实时抓取用户的账号密码。

Web 系统漏洞也称脆弱性，是 Web 系统各组成部分在硬件、软件和协议的具体实现或者安全策略上存在的缺陷和不足。2011 年，国家信息安全漏洞共享平台（China National Vulnerability Database，CNVD）共收集整理并公开发布信息安全漏洞 5547 个，较 2010 年增加 60.9%，其中高危漏洞有 2164 个，较 2010 年增加约 2.3 倍。在所有漏洞中，涉及各种应用程序的最多，占 62.6%。同时，利用漏洞对 Web 系统的攻击事件数量大增。

根据 Web 系统的结构，其安全漏洞通常归纳为 Web 应用程序漏洞、中间件漏洞、数据库漏洞以及主机服务器漏洞。

（1）Web 应用程序漏洞：具体指提供服务的 Web 应用程序中存在的安全性漏洞，这些问题大都是因为 Web 应用程序的编码存在一定的缺陷而引起。典型的 Web 应用程序漏洞包括跨站脚本、跨站请求伪造、SQL 注入和弱口令等。

（2）中间件漏洞：为 Web 应用程序提供服务的 Web 服务软件的安全漏洞，无论是 IIS、WebLogic、Apache 还是 Tomcat 都存在漏洞，需要不断地打补丁升级，常见的漏洞包括文件上传漏洞和文件包含漏洞等。

（3）数据库漏洞：为 Web 应用程序提供数据服务的应用系统漏洞，包括数据库提权漏洞、SQL 注入漏洞和弱口令等。

（4）主机服务器漏洞：操作系统以及运行在操作系统之上的应用的安全漏洞，如 Window/Linux/Unix 等操作系统本身的漏洞。

早期的 Web 站点仅仅提供静态 HTML 页面浏览服务，不同的用户从 Web 站点上获取到的信息也是完全相同的。在这种情况下，Web 站点受到的安全威胁仅仅与主机服务器漏洞有关。而目前 Web 系统提供的服务与早期服务相比要复杂得多，绝大部分 Web 应用程序提供的服务是动态解析生成的，包括用户注册、登录、查询、统计分析等服务，而这些用户专属服务需要在客户端和服务器之间进行双向数据交互才能完成，交互的数据中不乏个人隐私数据和金融信息。但是由于以下技术和管理方面的原因，Web 应用程序和主机服务器等存在大量漏洞。

（1）缺乏安全意识。很多 Web 应用程序的开发人员对 Web 应用安全领域相关的核心概念并不重视，甚至没有相关的安全编码概念，因此缺乏应对这些安全问题的技术与经验。

（2）设计缺陷多样。很多 Web 应用程序为节约时间和成本，由自己的内部员工或者外包给第三方中小企业完成，加之滥用第三方插件，导致 Web 应用程序都存在着设计缺陷。

（3）安全威胁更新快。随着 Web 技术的发展，新型的 Web 应用安全威胁出现得非常迅速，虽然在 Web 应用程序开发时处理了相关安全威胁，但是很可能在开发完成后会面临许多新的威胁。

（4）缺乏高质量的安全测试。受到开发成本与时间限制，很多开发团队只通过快速渗透测试发现明显的安全漏洞，甚至没有处理安全漏洞，从而造成安全隐患。

（5）主机安全配置不当。主机服务器的操作系统或者中间件都可能由于配置不当而存在安全风险，如日志管理配置可能引起系统或软件日志的丢失或泄漏、远程登录的配置不当可能引起非法用户的远程访问等。

（6）主机服务器代码问题。操作程序代码量非常庞大，如 Linux 操作系统的代码量已经超过了 1000 万行，Windows 操作系统的汇编代码也已经超过了 500 万行，而 Apache、Tomcat、IIS 等中间件的代码规模也是不容忽视的。庞大的程序规模必然导致安全问题的存在。如由于代码编写过程中的内存管理问题而导致的缓冲区溢出漏洞，从而引起远程代码执行；由于应用程序编写不当、数据过滤不严格而造成的代码注入，从而引起信息泄漏、验证绕过、远程代码执行等。

五、Web 系统安全威胁

由于运行在服务器端的 Web 应用程序无法干预浏览器的操作，攻击者可以篡改客户端与服务器间的交互数据，或者通过构造恶意参数，并利用 Web 系统存在的漏洞，使 Web 系统出现无法预料的错误或者异常状况，威胁到系统的安全性和可用性。

随着 Web 应用技术的发展和研究的深入，Web 系统所面临的安全威胁也越来越多。开放式 Web 应用程序安全项目（Open Web Application Security Project，OWASP）被众多权威性机构（如美国联邦贸易委员会、美国国防部、国际信用卡数据安全技术 PCI 标准等）列为 Web 应用程序安全规范。OWASP 专注于 Web 安全，它的十大最重要的 Web 应用程序威胁报告能够很好地反映 Web 安全所面临的威胁和这些威胁的发展趋势。从 2013 年的数据来看，Web 系统面临的安全形势依然严峻，其中以注入攻击、错误认证以及跨站脚本攻击尤为严重，具体的数据见表 1-2。

表 1-2　　2013 年 OWASP 十大 Web 系统安全威胁

A1	注入攻击	A6	关键数据暴露
A2	错误的认证和会话管理	A7	缺少功能层面的访问控制
A3	跨站脚本攻击	A8	跨站请求伪造
A4	不安全的对象直接引用	A9	应用已知脆弱性的组件
A5	安全配置错误	A10	未验证的重定向和传递

（1）注入攻击。由于应用程序缺少对输入数据的合法性检查，当攻击者把包含恶意指令的数据发送给应用程序解释器，诱使命令解释器执行非法命令或者执行未被授权的操作。常见的注入包括 SQL 注入、操作系统（Operating System，OS）注入、LDAP 注入等。其中，SQL 注入尤为常见，危害也最为严重，通过 SQL 注入攻击可以窃取或者篡改整个数据库的信息，甚至能够获得管理员级别的访问权限。SQL 注入攻击在攻击行为上和正常调用 Web 应用没有任何显著的不同，普通防火墙设备很难监测其攻击行为。SQL 注入攻击报告的案例非常多：2012 年，乌云网站曝出 12306 网站存在多个 SQL 注入漏洞；2011 年 12 月，黑客利用 SQL 注入漏洞攻击了国内最大的程序员社区网站 CSDN，造成大量用户数据泄漏。

（2）错误的身份认证和会话管理。身份认证和会话管理的方式包括提交用户名、密码、验证码、证书等。会话管理用于从大量无状态的 HTTP 连接中识别特定的用户。一般情况下，身份认证和会话管理都是同时使用的。身份认证的结果是获得一个令牌，并放在会话的 Cookie 中，之后就通过这个授权对用户身份进行识别，无需每次都要登录。如果 Web 应用程序在设计编码的过程中未对相关部分进行严格要求，身份认证和会话管理就不能够正确实现，攻击者通过窃听用户访问 Web 应用程序时的用户名、密码及会话（Session）数据，可以得到会话标识，进而冒充合法用户发起 HTTP 访问。如网上商店应用程序支持 URL 重写，把会话标识放在 URL 中：http://www.bookshop.com/items?sessionID=1234567?name=java。该网站的一个经过认证的用户，将该链接发给他的一个朋友，希望他朋友了解该商品信息，却不知道自己已经将自己的会话标识泄漏出去。他的朋友或者获得该会话标识的攻击者通过该链接，可以盗取其账号，甚至信用卡信息。

（3）跨站脚本攻击。当 Web 应用程序向浏览器发送没有经过验证和转码的不信任数据时，被攻击者利用，使得攻击者能够在受害者的浏览器上运行恶意脚本，从而劫持会话，或者转向恶意网站，发动跨站点脚本攻击。跨站点脚本攻击的主要危害是使得攻击者能够在受害人的浏览器上运行恶意脚本，从而危害受害人的数据安全或系统安全。XSS 根据攻击手段分为反射型、保存型和基于 DOM 的跨站点脚本攻击。跨站点脚本攻击对于静态网页不会产生任何影响，但是现在 Web 系统包含了大量的动态脚本用以提高用户体验，而这些系统如果不能很好地进行安全方面的设计就会受到此类攻击。根据 wooyun.org 公布的漏洞列表，2011 年 6 月 28 日晚，新浪微博出现了一次比较大的 XSS 攻击事件，微博用户会自动向自己的粉丝发送含病毒私信和微博，有人单击后会再次中毒，形成恶性循环。2014 年 3 月 9 晚，百度贴吧出现跨站点脚本攻击事件，六安吧等几十个贴吧出现单击推广贴会自动转发，并且受到 XSS 攻击的转帖在吧友所关注的每个贴吧都会转一遍，病毒循环发帖。此外，谷歌（Google）、微软公司的 xbox360、Twitter、Facebook、MySpace、Orkut、新浪微博网站上，雅虎公司基于 Web 的电子邮件服务上也均爆出过跨站点脚本攻击漏洞。

（4）不安全的对象直接引用。指一个已经授权的用户通过更改访问时的一个参数，从而访问到原本其并没有得到授权的对象。Web 应用往往在生成 Web 页面时会用它的真

实名字，且并不会对所有的目标对象访问时来检查用户权限，这就造成了不安全的对象直接引用的漏洞。

（5）安全配置错误。安全配置错误可以发生在一个应用程序堆栈的任何层面，包括中间件、Web 服务器、数据库服务器、数据库、Web 应用程序等。攻击者通过访问默认账户、未使用的网页、未安装补丁的漏洞、未被保护的文件和目录等，以获得对系统未授权的访问。如 Web 服务器管理控制台默认安装，没有修改用户名和密码，攻击者就可以利用默认用户名和密码登录该系统，最终完全控制服务器主机。

（6）关键数据暴露。保护与加密敏感数据已经成为网络应用最重要的组成部分。最常见的漏洞是应该进行加密的数据没有进行加密。使用加密的情况下，常见问题是不安全的密钥和使用弱算法加密。攻击者可能会窃取或篡改这些弱保护的数据以进行信用卡诈骗、身份窃取或其他犯罪。敏感数据值需额外的保护，比如在存放或传输过程中的加密，以及在与浏览器交换时进行特殊的预防措施。

（7）缺少功能层面的访问控制。有时功能级的保护是通过系统配置管理的，当系统配置错误时，开发人员必须做相应的代码检查，做到在每个功能被访问时在服务器端执行相同的访问控制检查。否则应用程序不能进行正确的保护页面请求，攻击者就可利用这种漏洞访问未经授权的功能模块。

（8）跨站请求伪造。也称为会话叠置，它是一种会话劫持攻击，强迫受害者的浏览器向一个易受攻击的 Web 应用程序发送请求，最后达到攻击者所需要的操作行为，分为本站点请求伪造和跨站点请求伪造两种。本站点请求伪造常常与保存型 XSS 漏洞结合使用；跨站点请求伪造通过强迫受害者的浏览器向一个存在漏洞的 Web 应用程序发送请求来达到攻击者的预期目的。

（9）应用已知脆弱性的组件。开发人员使用的组件也会含有漏洞，这些漏洞能够被自动化工具发现和利用。如果一个带有漏洞的组件被利用，这种攻击可以造成更为严重的数据丢失或服务器接管。应用程序使用带有已知漏洞的组件会破坏应用程序防御系统，并使一系列可能的攻击和影响成为可能。

（10）未验证的重定向和传递。在重定向和转发中极为普遍，如果重定向或转发的 URL 中带有未经验证的用户输入参数，攻击者可以重定向受害用户到钓鱼软件或恶意网站，或者使用户去访问未授权的页面。

第二节　Web 系统安全技术

针对 Web 系统自身的脆弱性和来自外部的安全威胁，防护措施主要包括网络层安全防护、应用层安全防护和数据库层安全防护。如图 1-5 所示，主要的安全防护技术和工具包括网络防火墙、入侵检测系统（IDS）、入侵防御系统（IPS）、Web 应用防火墙、安全审计技术、代码安全技术、数据加密技术和数据库安全技术。

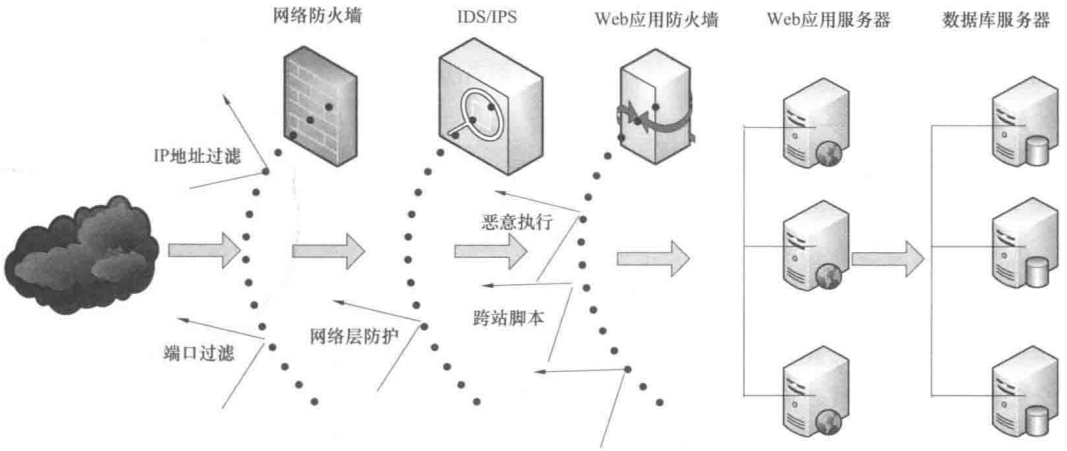


图 1-5　Web 系统安全防护示意图

一、网络层安全防护

1. 防火墙

防火墙是位于两个或多个网络间实施网间访问控制的一组软硬件的集合，通常处于内部局域网与 Internet 之间，起到一个安全网关的作用，限制 Internet 用户对内部网络的访问及管理内部用户访问 Internet 的权限，从而保护内部网络免受非法入侵。防火墙是一种被动的技术，它假设了网络边界的存在，对内部的非法访问难以有效地控制。

防火墙技术作为目前实现 Web 系统外围网络防护安全的一种手段，其主要功能包括：

（1）访问控制。限制未经授权的用户访问内部信息系统和资源，拒绝未经授权的用户存取敏感数据和信息，同时允许合法用户不受妨碍地访问网络资源。

（2）安全审计。防火墙可以对内、外部网络存取和访问进行监控审计。防火墙能就所有访问进行日志记录。当发生攻击或可疑动作时，防火墙能进行适当的报警，并提供网络是否受到攻击的详细审计信息。

（3）策略管理。防火墙可设置统一的安全方案配置，将所有安全软件（如口令、加密、身份认证、审计等）配置在防火墙上。相比安全问题分散到各个主机上，防火墙的集中安全管理更经济。

（4）双向地址转化。提供 IP 地址转换和 IP 及 TCP/UDP 端口映射，实现 IP 复用和隐藏网络结构，解决 IP 地址空间不足的问题，同时隐藏内部网的结构，强化内部网的安全。

网络防火墙可以在很大程度上提高 Web 系统的安全性能，但是防火墙不能解决所有的安全问题，防火墙也有其自身的局限性。防火墙的不足主要体现在以下几方面。

（1）传统的防火墙作为访问控制设备，主要工作在 OSI 模型的第三层和第四层，进行基于 IP 报文的检测，无法理解 HTTP 会话语言，不能对 Web 系统客户端的输入进行验证。对于针对高层的合理访问攻击如 SQL 注入、跨站攻击等，无有效手段。

（2）防火墙不能防止来自内部网络的攻击和内部的数据泄密。防火墙对于来自内部的网络攻击无能为力，同时也不能够防范内部用户的主动泄密。