活页式教材

校企“双元”合作新形态教材

Web 应用安全与防护

主编　常云刚　李超强
主审　黎乾坤

西北工业大学出版社

Web
应用安全与防护

ISBN 978-7-5775-0303-5

定价:42.00元

校企“双元”合作新形态教材

Web 应用安全与防护

主　编　常云刚　李超强
副主编　李光蔓　包昌权
主　审　黎乾坤

西北工业大学出版社
西　安

【内容简介】 本书共9个项目，分别是Web安全基础、登录认证、数据库交互、用户输入、个人信息修改、文件上传、文件包含、移动设备Web应用的防护和Web防护体系建设。

本书适合作为高等职业教育信息安全技术应用、计算机网络技术、软件技术等相关专业的教材，也可以作为对网络安全感兴趣的渗透测试人员、Web开发人员等的自学参考书。

图书在版编目(CIP)数据

Web应用安全与防护 / 常云刚，李超强主编. 西安 ：西北工业大学出版社，2026. 2. -- ISBN 978-7-5775-0303-5

Ⅰ. TP393.08

中国国家版本馆CIP数据核字第2026K5R025号

Web YINGYONG ANQUAN YU FANGHU

Web 应 用 安 全 与 防 护

常云刚　李超强　主编

责任编辑：孙　倩　　**装帧设计**：张瑞阳

责任校对：朱辰浩

出版发行：西北工业大学出版社

通信地址：西安市友谊西路127号　　邮编：710072

电　　话：(029)88491757，88493844

网　　址：www.nwpup.com

印 刷 者：大厂回族自治县聚鑫印刷有限责任公司

开　　本：787 mm×1 092 mm　　1/16

印　　张：10.25

字　　数：231千字

版　　次：2026年2月第1版　　2026年2月第1次印刷

书　　号：ISBN 978-7-5775-0303-5

定　　价：42.00元

前言

随着互联网的迅猛发展，Web 应用已经成为现代社会中不可或缺的一部分。无论是在电子商务、在线服务、社交媒体领域还是在政府机构办公领域，Web 应用都扮演着至关重要的角色。然而，随着 Web 应用的普及和深入，其面临的安全威胁也日益严峻。黑客攻击、数据泄露、恶意软件注入等安全问题层出不穷，给企业和个人带来了巨大的经济损失和声誉损害。

在这样的背景下，Web 应用安全技术显得尤为重要。本书旨在为读者提供一本全面、深入、实用的 Web 应用安全指南，不仅介绍了 Web 应用安全的基础知识和常见威胁，还深入探讨了各种安全技术和防御策略，帮助读者掌握构建安全的 Web 应用环境的方法。

本书结构清晰，逻辑严密，首先介绍了 Web 技术的发展历程，让读者对 Web 技术有初步的认识。接着详细分析了 Web 应用面临的各种威胁，包括 SQL 注入攻击、跨站脚本攻击、跨站请求伪造等，揭示了这些威胁的攻击原理、危害程度和防御方法。在深入分析威胁的基础上，进一步探讨了 Web 应用安全的核心技术，包括安全编码实践、输入验证、会话管理、访问控制等关键技术，这些技术是构建安全 Web 应用的基础。同时，本书还介绍了 Web 应用防火墙、入侵检测系统、安全审计等安全工具的使用，这些工具可以帮助读者更好地监控和防御 Web 应用的安全威胁，建立全面的 Web 应用安全管理体系。

本书由云南财经职业学院常云刚、李超强担任主编，云南世纪慧铭集团有限公司李光蔓、包昌权担任副主编，云南财经职业学院杨明、周林莉、杨文庆、杨喆和云南世纪慧铭集团有限公司刘彬、杨光明参与编写，全书由云南财经职业学院黎乾坤教授主审。具体编写分工如下：常云刚编写项目 1 和项目 2，李超强

编写项目 3 和项目 4，李光蔓和包昌权编写项目 5，杨明编写项目 6，周林莉编写项目 7，杨文庆和杨喆编写项目 8，刘彬和杨光明编写项目 9。在编写本书过程中，笔者参阅了许多文献与资料，在此向其作者表示感谢。

由于笔者水平有限，书中难免存在不足之处，敬请广大读者批评指正。

编　者

2025 年 9 月

目录

项目 1

Web 安全基础

项目学习重点

1. 了解 Web 技术的发展历程；
2. 了解 Web 安全的核心问题；
3. 学习 HTTP 及其安全性；
4. 了解 Web 应用中的编码与加密；
5. 熟知 Web 1.0、Web 2.0、Web 3.0 的优缺点及区别。

在本项目中，通过学习登录认证的工作原理，读者可以了解登录认证过程中出现安全问题的原因，并结合常见工具的使用来了解登录认证漏洞的原理，掌握登录认证漏洞的检测方法和测试技巧，以及漏洞的修复方法。

任务 1.1 了解 Web 技术的发展历程

Web 的发展历程可以被划分为三个阶段，即 Web 1.0、Web 2.0 和 Web 3.0，每个阶段都对网络技术的发展和应用产生了深远的影响。

1.1.1 Web 1.0

Web 1.0 是个人计算机时代的互联网，用户利用 Web 浏览器通过门户网站单向获取内容，主要进行浏览、搜索等操作。在这个阶段，用户只是被动接受内容，没有互动体验。

Web 1.0 时代是一个群雄并起、逐鹿网络的时代，虽然各个网站采用的手段和方法不同，但第一代互联网有诸多共同的特征，表现在技术创新主导模式、基于点击流量的盈利共通点、门户合流、明晰的主营兼营产业结构、动态网站。在 Web 1.0 上做出巨大贡献的公司有 Netscape、Yahoo 和 Google。Netscape 研发出第一个大规模商用的浏览器，Yahoo 的杨致远提出了互联网黄页，而 Google 后来居上，推出了大受欢迎的搜索服务。

Web 1.0 具有如下特点。

（1）Web 1.0 基本采用的是技术创新主导模式，信息技术的变革和使用对于网站的新生与发展起到了关键性的作用。新浪以技术平台起家，搜狐以搜索技术起家，腾讯以即时通信技术起家，盛大以网络游戏起家，在这些网站的创始阶段，技术性的痕迹很重。

（2）Web 1.0 的盈利都基于一个共通点，即巨大的点击流量。无论是早期融资还是后期获利，依托的都是为数众多的用户和高点击率，以点击率为基础上市或开展增值服务，是受众的基础，决定了盈利的水平和速度，充分地体现了互联网的眼球经济色彩。

（3）Web 1.0 的发展出现了向综合门户合流现象，早期的新浪、搜狐、网易等，继续坚持了门户网站的道路，而腾讯、MSN、Google 等网络新贵，都纷纷走向了门户网络，尤其是对于新闻信息，有着极大的、共同的兴趣。这一情况的出现，使门户网站本身的盈利空间更加广阔，盈利方式更加多元化，占据网站平台，可以更加有效地实现增值意图，并延伸至主营业务之外的各类服务。

（4）在 Web 1.0 合流的同时，还形成了主营业务与兼营业务相结合的明晰产业结构。新浪以新闻 + 广告为主，网易拓展游戏，搜狐延伸门户矩阵，各家以主营业务作为突破口，以兼营业务作为补充点，形成拳头加肉掌的发展方式。

（5）Web 1.0 不以 HTML 为语言，在 Web 1.0 时代，动态网站已经广泛应用，如论坛等。

1.1.2 Web 2.0

Web 2.0 是相对于 Web 1.0 的新时代，指一个利用 Web 的平台，由用户主导而生成的内容互联网产品模式，为了区别传统由网站雇员主导生成的内容而定义为第二代互联网，即 Web 2.0。

Web 2.0 的概念始于奥莱利（O'Reilly）公司和 MediaLive International 公司之间的一场头脑风暴论坛。作为互联网先驱的 O'Reilly 副总裁，戴尔·多尔蒂（Dale Dougherty）指出，伴随着令人激动的新程序和新网站间惊人的规律性，互联网不仅远没有“崩溃”，甚至比以往更重要。更进一步说，那些得以活过泡沫破裂的公司之间似乎拥有某种相同点。“Web 2.0”的概念由此诞生了。

Web 2.0 可以说是信息技术发展引发网络革命所带来的面向未来、以人为本的创新 2.0 模式在互联网领域的典型体现，是对由专业人员织网到所有用户参与织网的创新民主化进程的生动注释。

与 Web 1.0 相比，Web 2.0 具有如下特点。

（1）用户参与网站内容制造。与 Web 1.0 网站单项信息发布的模式不同，Web 2.0 网站的内容通常是由用户发布的，这使得用户既是网站内容的浏览者也是网站内容的制造者，也就意味着 Web 2.0 网站为用户提供了更多参与的机会。例如，博客网站和维基（Wiki）就是用户创造内容的典型代表，而用户设置标签技术将传统网站中的信息分类工作直接交给用户来完成。

（2）Web 2.0 更加注重交互性。Web 2.0 不仅实现了用户在发布内容的过程中与网络服务器之间的交互，而且实现了同一网站不同用户之间的交互，以及不同网站之间信息的交互。

（3）符合 Web 标准的网站设计。Web 标准是国际上正在推广的网站标准，通常所说的 Web 标准一般是指网站建设采用基于 XHTML 的网站设计语言，实际上，Web 标准并不是某一标准，而是一系列标准的集合。Web 标准中典型的应用模式是“CSS+XHTML”，摒弃了 HTML 4.0 中的表格定位方式，其优点之一是网站设计代码规范，并且减少了大量代码，减少网络带宽资源浪费，加快了网站访问速度。更重要的一点是，符合 Web 标准的网站对于用户和搜索引擎更加友好。

（4）Web 2.0 网站与 Web 1.0 没有绝对的界限。Web 2.0 技术可以成为 Web 1.0 网站的工具，一些在 Web 2.0 概念之前诞生的网站本身也具有 Web 2.0 特性。例如，B2B 电子商务网站的免费信息发布和网络社区类网站的内容也来源于用户。

（5）Web 2.0 的核心不是技术而在于指导思想。Web 2.0 有一些典型的技术，但技术是为了达到某种目的所采取的手段，因此 Web 2.0 技术本身不是 Web 2.0 网站的核心，重要的在于典型的 Web 2.0 技术体现了具有 Web 2.0 特征的应用模式。因此，与其说 Web 2.0 是互联网技术的创新，不如说是互联网应用指导思想的革命。

（6）Web 2.0 是互联网的一次理念和思想体系的升级换代，由原来的自上而下的由少数资源控制者集中控制主导的互联网体系，转变为自下而上的由广大用户集体智慧和力量主导的互联网体系。

1.1.3　Web 3.0

Web 3.0 是对 Web 2.0 的改进，在此环境下，用户不必在不同中心化的平台创建多重身份，而是能打造一个去中心化的通用数字身份体系通行于各个平台。 Web 3.0 被用来描述互联网潜在的下一阶段，一个运行在区块链技术之上的“去中心化”的互联网。

Web 3.0 概念于 2014 年提出，定位用户互联网，为去中心化模式。Web 3.0 是开源协议，但通过密码经济学实现集体所有；独立于传统组织，代码按规定执行；重视开源软件、用户对数据的所有权以及无许可访问，创造一个共同的身份和协作意识。Web 3.0 的代表应用有比特币、以太坊等，相关核心技术包括区块链等。

1. Web 3.0 的技术架构

Web 3.0 的技术架构可分为基础层、平台层、应用层。基础层技术由区块链这一融合技术构成，包括分布式账本、共识算法、密码学技术、智能合约、分布式存储、跨链等；平台层技术包含人工智能、大数据、扩展现实、云计算、渲染与 3D 建模等技术；应用层技术类别更多、范围更广，同时也存在更多的可能性，各项技术的融合发展具有巨大的潜在空间。Web 3.0 同时兼容 Web 1.0 和 Web 2.0 的区块链技术，赋予了 Web 3.0 去中心化、开放、独立、安全等特点。

2. Web 3.0 的优势

（1）数据主权归还用户。过去，Web 2.0 平台掌握着用户的数据，而现在每一个用户能自己掌握数据，不会再因为平台的关停而丢失数据。

用户的数据具备价值，但 Web 2.0 平台不会把价值分给用户，反而利用用户数据进行商业变现，如广告的精准投放、某些平台的数据分析报告贩售等，而这一切授权都被包含在用户注册时的“用户协议”中，Web 2.0 平台静悄悄地拿走了用户数据的归属权利。

（2）跨平台的身份认证。过去用户无法实现跨平台（不同集团的平台）的账号操作，在每一个平台都需要注册一个账号才可以操作，且各个平台之间往往无法直接互通。例如，支付宝上的余额无法直接转到微信钱包上，需要通过银行卡来进行周转。

在 Web 3.0 平台，人人皆可以拥有一个跨平台的唯一性身份认证，并记录在链上的所有数据与行为，一个身份全网使用，大大提高了用户使用效率及便捷性。

（3）互联网的价值重构。过去互联网平台 / 产品最重要的就是用户及用户数据，因此新产品发布时都会尽可能地用低价、免费来吸引用户，以获取用户数的累积。Web3.0 的互联网从“平台垄断”转向“用户主权”，从“数据剥削”转向“价值共享”。用户不仅创造内容，还能通过智能合约、NFT（非同质化代币）等方式，直接参与平台价值的分配。

3. Web 3.0 的弊端与反思

（1）用户仍需要 Web 2.0 平台。普通用户没有直接与区块链交互的能力，必须仰赖 Web 2.0 的平台。例如，交易 NFT 需要通过 OpenSea 等一众 NFT 交易平台来进行交易，普通用户无法成为区块链中真正普惠收益的一方，依然需在平台的掌控下操作。

（2）去中心化的本质，很可能是“再造中心化”。虽然从表面上看整个 Web 3.0 是采用区块链实现的去中心化世界，将各平台、应用的中心化权利释放，归还给用户，但在实际运作中，各项目仍然是发起人团队掌握三四成的份额，对 DAO（去中心化自治组织）有着控制权，同时，底层区块链平台（如以太坊）也掌握着协议制定与升级的主导权，形成对个别链上的小组织的去中心化，掌握链上核心权利及资源的少数阶层成为超级中心。

（3）安全性问题。目前 Web 3.0 主要的身份认证方式为“加密钱包”，但当加密钱包遇到“提示词”泄露时，可能会导致虚拟资产被盗卖、盗取。

（4）加密货币不稳定。比特币、以太币等一众加密货币和各类 NFT 项目，因内部

“去中心化”并无明确的货币 / 价格政策调控的能力，导致交易结果一切以市场波动和项目发起方的情况为主，不时就会遇到价值暴跌的情况；并且交易时间无限制，为 24 h 持续交易，可能一觉醒来，身价就瞬间没有了。

（5）金融监管缺失。加密货币与 NFT 交易也存在着大量的金融监管缺失问题，成为全球最新的洗钱管道，使用多人多加密钱包的形式，有组织地操作价值之间的流转，很容易就能将钱洗白或洗出境内，逃离境内的管制与追查。

总的来说，Web 3.0 将用户数据主权归还用户，并打破原有的用户与平台 / 产品耦合在一起的账号关系，用户数据不再是互联网平台 / 产品的核心资产与“护城河”，互联网平台 / 产品价值则将重构成其他的形式。

任务 1.2　了解 Web 安全的核心问题

随着互联网技术的飞速发展，Web 应用已成为现代生活的重要组成部分。然而，随之而来的安全风险也日益严峻。Web 安全的核心问题涉及多个方面，主要包括用户输入不可信、结构查询语言（structured query language，SQL）注入攻击、跨站脚本（cross site script，简称 XSS）攻击、跨站请求伪造（cross site request forgery，CSRF）、分布式拒绝服务（distributed denial of service，DDoS）攻击、域名系统（domain name system，DNS）劫持风险以及 JaveScript 对象表示法（JSON）数据劫持等。下面将对这些问题进行详细探讨。

1.2.1　用户输入不可信

用户输入不可信是 Web 安全中的首要问题。Web 应用通常依赖用户输入进行各种操作，如搜索、登录、注册等。然而，恶意用户可能会尝试输入恶意数据，以触发系统漏洞或进行攻击。因此，对用户输入进行严格验证和过滤至关重要。通过限制输入字符集、使用安全的编码方案（如 HTML 实体编码）以及实施服务器端验证，可以有效降低恶意输入带来的风险。

1.2.2　SQL 注入攻击

SQL 注入攻击是一种常见的 Web 安全漏洞，攻击者通过构造特殊的 SQL 语句，尝试插入、更新或删除数据库中的数据。这种攻击通常发生在应用未对用户输入进行充分验证的情况下。为了防止 SQL 注入攻击，应采取以下措施。

（1）使用预编译的 SQL 语句。

（2）避免在 SQL 语句中直接拼接用户输入。

（3）根据最小权限原则对数据库用户权限进行配置。

（4）定期检查数据库日志和审计记录，以发现潜在攻击。

1.2.3 XSS 攻击

XSS 允许攻击者在受害者的浏览器中执行恶意脚本。这种攻击通常通过插入恶意脚本到 Web 页面中实现，从而窃取用户信息、会话令牌或执行其他恶意操作。为了防止 XSS 攻击，应采取以下措施。

（1）对用户输入进行严格的输出编码。

（2）设置合适的 HTTP 头部（如 Content-Type 和 X-Content-Type-Options）。

（3）使用安全的内容安全策略（CSP）。

（4）定期进行代码审查和安全测试。

1.2.4 CSRF

CSRF 允许攻击者诱使受害者在不知情的情况下执行恶意请求。这种攻击通常通过利用受害者的会话令牌和权限来发起恶意操作。为了防止 CSRF 攻击，应采取以下措施。

（1）使用抗 CSRF 令牌（如双因素认证）。

（2）对敏感操作进行二次确认。

（3）实施严格的会话管理策略。

（4）定期检查和更新 Web 应用的权限控制机制。

1.2.5 DDoS 攻击

DDoS 通过向目标服务器发送大量伪造的网络流量，以耗尽其资源并导致服务中断。这种攻击通常由多个攻击者协同发起，以扩大攻击规模。为了防止 DDoS 攻击，应采取以下措施。

（1）部署高性能的防火墙和入侵检测系统（intrusion detection system，IDS）。

（2）与网络服务提供商合作，建立 DDoS 防护机制。

（3）使用流量清洗和分散技术，以降低攻击效果。

（4）定期进行压力测试和性能评估，以确保系统具备足够的应对能力。

1.2.6 DNS 劫持风险

DNS 劫持是一种攻击手段，攻击者通过篡改 DNS 记录，将用户重定向到恶意网站或服务器。这种攻击可能导致信息泄露、数据窃取或身份盗窃等严重后果。为了防止 DNS 劫持风险，应采取以下措施。

（1）使用安全的 DNS 解析服务（如 DNSSEC）。

（2）定期检查 DNS 记录，确保其完整性和准确性。

（3）实施严格的访问控制和权限管理策略。

（4）对用户进行安全教育和培训，增强用户的安全意识。

1.2.7 JSON 数据劫持

JSON 数据劫持是一种攻击手段，攻击者通过利用 JSON 数据解析漏洞，尝试窃取或篡改敏感数据。这种攻击通常发生在 Web 应用未对 JSON 数据进行适当验证和防护的情况下。为了防止 JSON 数据劫持，应采取以下措施。

（1）对 JSON 数据进行严格的输入验证和输出编码。

（2）使用安全的 JSON 解析库和工具。

（3）实施 CSP，以防止跨站脚本攻击。

（4）定期对 Web 应用进行代码审查和安全测试，以发现潜在漏洞。

为了保障 Web 应用的安全性，应采取多种措施进行防护和应对。同时，定期对 Web 应用进行安全评估和改进也是至关重要的。只有通过综合采取多种技术手段和管理措施，才能有效地降低 Web 应用面临的安全风险。

任务 1.3　了解 HTTP 及其安全性

目前较流行的用户 Web 协议有超文本传送协议（hypertext transfer protocol，HTTP）与无线应用协议（wireless application protocol，WAP）。HTTP 对应的安全协议是超文本传输安全协议（hypertext transfer protocol secure，HTTPS），无线应用协议包括无线传输层安全协议（wireless transport layer security，WTLS）和 Wi-Fi 网络安全存取协议（Wi-Fi protected access，WPA）。

1.3.1 HTTP 概述

HTTP 是互联网上应用最为广泛的一种网络协议，所有 WWW 文件都必须遵守这个标准。设计 HTTP 最初的目的是提供一种发布和接收 HTML 页面的方法。

HTTP 的发展是万维网协会（world wide web consortium）和 Internet 工作小组（Internet engineering task force）合作的结果，它们最终发布了一系列的征求意见稿（request for comments，RFC），其中最著名的就是 RFC 2616。

HTTP 是一个客户端和服务器端请求和应答的标准，客户端是终端用户，服务器端是网站。通过使用 Web 浏览器、网络爬虫或者其他工具，客户端发起一个到服务器上指定端口（默认端口为 80）的 HTTP 请求。这个客户端称为用户代理（user agent）。应答的服务器上存储着一些资源，如 HTML 文件和图像。这个应答服务器称为源服务器（origin

server）。在用户代理和源服务器中间可能存在多个中间层，如代理、网关，或者隧道（tunnel）等。尽管 TCP/IP 是互联网上最流行的应用协议，HTTP 并没有规定必须使用它和基于它支持的层。事实上，HTTP 可以在任何其他互联网协议上，或者在其他网络上实现。HTTP 只假定其下层协议提供可靠的传输，任何能够提供这种保证的协议都可以被其使用。

通常，由 HTTP 客户端发起一个请求，建立一个到服务器指定端口（默认端口是 80，其他端口也可以）的 TCP 连接。HTTP 服务器则在该端口监听客户端发送过来的请求。一旦收到请求，服务器向客户端发回一个状态行（如“HTTP/1.1 200 OK”）和响应的消息，消息的消息体可能是请求的文件、错误消息，或者其他一些信息。

HTTP 使用 TCP 而不是 UDP 的原因在于打开一个网页必须传送很多数据，而 TCP 提供传输控制、按顺序组织数据和错误纠正。在互联网或其他网络上，这并不妨碍 HTTP 应用在其他协议的顶端。HTTP 仅仅期望可靠的传输，任何提供这种保证的协议都可以使用。

通过 HTTP 或者 HTTPS 请求的资源由统一资源标识符（uniform resource identifier，URI）来标识。

HTTP 是一种通用的、不分状态的协议，除了诸如名称服务和分布对象管理系统之类的超文本用途外，还可以通过扩展它的请求方式、错误代码和报头来完成许多任务。HTTP 的一个特点是数据表示方式的典型性和可协商性允许独立于传输数据而建立系统。在 1990 年 WWW（全球信息网）刚刚起步的时候，HTTP 就得到了应用。HTTP 的第一个版本叫作 HTTP 0.9，是一种为互联网原始数据传输服务的简单协议。由 RFC 1945 定义的 HTTP 1.0 进一步完善了这个协议，它允许消息以类似多用途互联网邮件扩展（multipurpose internet mail extensions，MIME）的格式传送，包括有关数据传输的维护信息和关于请求 / 应答的句法修正。但是，HTTP 1.0 没有充分考虑到分层代理，高速缓存的作用以及对稳定连接和虚拟主机的需求。随着不完善的进程应用的激增，HTTP 1.0 迫切需要一个新的版本，以便使两个通信应用程序能够确定彼此的真实性能。于是 RFC 2616 定义了 HTTP 中一个现今被广泛使用的版本——HTTP 1.1。

HTTP 也是用于用户代理之间及代理 / 网关到其他网络系统的通用通信协议，这样的网络系统可能由 SMTP、NNTP、FTP、Gopher 和 WAIS 协议支持。这样，HTTP 允许不同的应用程序对资源进行基本的超媒体访问。

HTTP 是不安全的，攻击者可以通过监听和中间人攻击等手段获取网站账户和敏感信息等，所以很快推出了 HTTPS。

1.3.2 HTTP 的安全

虽然 HTTP 是一个标准，经过很多专家讨论过，但使用 HTTP 时，仍然需要注意一些可能的安全问题。

1. 敏感信息

HTTP 客户端经常会接触到大量的个人信息（如用户名、位置、邮件地址、密码等），这时要小心，不要通过 HTTP 泄露任何资源。

服务器是存储用户请求数据的地方，有些请求会被记录在日志中，用户需要保护好日志，以免日志被泄露。由于很多服务器、代理或者用户代理都可能会记录 Request-RUI 到日志或者其他地方，所以提交含有敏感信息的数据时，建议使用 POST 方法，而不是 GET 方法。

与一般的协议相比，HTTP 不能管理传输数据的内容，也不能提供任何方法决定任何请求的信息中是否有敏感的数据。因此，关于信息的控制的责任更多地落在了信息提供者的肩上。在这里着重强调四个头部字段：Server、Via、Referer 和 From。

（1）Server。如果服务器的软件的版本信息被泄露，那么可能是因为该版本有安全漏洞而导致更加容易受到攻击。目前大多数服务器的 Server 域是可以配置的，应尽量通过配置来隐藏软件的版本信息。

（2）Via。作为网络防火墙入口的代理，应该特别小心在防火墙后能识别主机的头部信息的传输。特别地，应该删除在防火墙后生成的 Via 域，或者替换成为净化的版本。

（3）Referer。Referer 头部字段允许被读取并且用于做反向链接，这确实非常有用，但如果用户的信息没有从 Referer 中分隔出来，就可能被滥用。即使用户信息被删除，但 Referer 域可能指向的是一个私有的文档的 URI，而这个文档是不应该公开的。客户端不应该在不安全的协议上传输 Referer。

（4）From。From 域中发送的信息可能与站点的安全策略相冲突，因此，如果用户不能控制这个域（禁用、启用或者修改域的内容），就不应该传输 From。用户可以设置这个域的内容为用户喜欢或者应用程序默认的配置的值。

2. 基于文件和路径的攻击

HTTP 源服务器的实现应该注意只返回被允许访问的文档。如果 HTTP 服务器直接将 URI 翻译成文件系统的调用，有可能导致预料之外的文件返回给客户端。例如，UNIX、Windows 和其他一些操作系统使用“..”指向当前目录的上一层路径，在这类系统中，HTTP 服务必须禁止 Request-URI 中含有这样的数据，否则可能导致 HTTP 服务器访问控制范围外的资源被访问。同样，只希望服务器内部引用的文件（如访问控制文件、配置文件和脚本代码等）必须严格保护从而不被意外访问。经验表明，这类 HTTP 服务器实现中的小问题往往会演变成大风险。

3.DNS 欺骗

使用 HTTP 的客户端严重依赖于域名服务，因此故意将 DNS 名和不相关的 IP 关联起来，可以发动 DNS 欺骗攻击。

特别是，HTTP 客户端依赖于自己的域名解析器来确认 IP 和 DNS 的匹配，而不是缓存以前查询的主机名称。不过，很多平台已经能够在本地查询缓存的内容。缓存是可行

的，但是，仅当域名服务器报告的生存周期（time to live，TTL）信息表明缓存有效时才可以。也就是说，如果 HTTP 客户端为了提高性能缓存主机名查询效果，那么它们必须观察 DNS 报告的 TTL 信息；否则，它们可能在以前所访问的服务器的 IP 地址改变时被欺骗。

4. Location 头欺骗

如果一个服务器支持多个组织，而且这些组织之间并不相互信任，那么服务器必须检查响应消息 Location 和 Content-Location 头，以防一个组织尝试访问一些它们没有被授权的资源。

5. 代理和缓存

依据代理和缓存的本质，因为 HTTP 代理是中间者，所以存在中间者攻击的机会。如果代理系统被攻破，将会严重威胁系统安全和用户隐私。代理可以访问到与用户相关的安全性方面的信息、个人信息、组织信息等。缺乏攻击抵抗力的代理可以被用于发动广泛的其他潜在攻击。

代理系统应该保护包含或者传输敏感信息的系统。特别是，代理收集的日志经常包含高度敏感的个人信息或者组织信息，应该小心保护日志不被泄露。

缓存又有额外的潜在弱点，因为缓存的内容对恶意攻击来说是具有吸引力的目标。因为缓存内容会在 HTTP 请求处理之后仍然存在，针对缓存的攻击能够泄漏用户认为已经删除了的缓存信息，所以缓存的内容应该被视为敏感信息来保护。明智地使用密码学，在适当的时候可以防止大范围的安全和隐私攻击。另外，对代理的拒绝服务攻击也存在，这也很难防范。

1.3.3 HTTPS 的安全性分析

HTTPS 是超文本传输协议和 SSL/TLS 的组合，用以提供加密通信及对网络服务器身份的鉴定。HTTPS 连接经常被用于万维网上的交易支付和企业信息系统中敏感信息的传输。注意，HTTPS 不应与在 RFC 2660 中定义的安全超文本传输协议（S-HTTP）混淆。

网景在 1994 年创建了 HTTPS，并将其应用在网景导航者浏览器中。最初，HTTPS 是与 SSL 一起使用的，在 SSL 逐渐演变到 TLS 时，最新的 HTTPS 也由在 2000 年 5 月公布的 RFC 2818 正式确定。

HTTPS 的主要思想是在不安全的网络上创建安全信道，并可在使用适当的加密套件和服务器证书可被验证和信任时，针对窃听和中间人攻击提供合理的保护。

HTTPS 的信任继承基于预先安装在浏览器中的证书颁发机构（如 VeriSign、Microsoft 等）（意为“我信任的证书颁发机构告诉我可以信任”）。因此，一个到某网站的 HTTPS 连接可被信任，当且仅当以下情况。

（1）用户相信他们的浏览器正确实现了 HTTPS 且安装了正确的证书。

（2）用户相信证书颁发机构仅信任合法的网站。

（3）被访问的网站提供了一个有效的证书，表示它是由一个被信任的证书颁发机构

签发的（大部分浏览器会对无效的证书发出警告）。

（4）该证书正确地验证了被访问的网站（例如，访问 https://ww.example.com 时收到了给“Example Inc.”而不是其他组织的证书）。

（5）互联网上相关的节点是值得信任的，或者用户相信本协议的加密层（TLS 或 SSL）不能被窃听者破坏。

HTTP 的 URL 由“http://”起始且默认使用端口 80，HTTPS 的 URL 由“https://”起始且默认使用端口 443。HTTPS 报文中的任何东西都被加密，包括所有报头和载荷。除了可能的选择密文攻击之外，一个攻击者所能知道的信息只有在两者之间有一个连接这一事实。虽然最近出现了 SSL Beast 攻击，但是，它是建立在使用老版本的 SSL/TLS 协议的基础之上，利用了老版本协议的一个漏洞，如果使用最新版本的安全协议，问题就自然解决了。

因为 SSL 在 HTTP 之下工作，对上层协议一无所知，所以 SSL 服务器只能为一个 IP 地址 / 端口组合提供一个证书。这就意味着在大部分情况下，使用 HTTPS 的同时支持基于名字的虚拟主机是很不现实的。一种称为服务器名称指示（server name indication，SNI）的方案通过在加密连接创建前向服务器发送主机名解决了这一问题。Firefox 2、Opera 8 和运行在 Windows Vista 的 Internet Explorer 7 都加入了对 SNI 的支持。

任务 1.4　了解 Web 应用中的编码与加密

字符是各种文字和符号的总称，包括各个国家文字、标点符号、图形符号、数字等。世界上存在大量不同的语言，每种语言所使用的文字或格式不同。在 Web 系统中，必须考虑使用某种编码方式来表现语言所对应的文字和格式。目前，常见的语言都有对应的字符编码，字符编码就是约定某个字在计算机中的编号。但不同的编码中，同一个字对应的编号完全不同，因此容易形成乱码。

1.4.1　针对字符的编码

字符编码有很多种类型，常用的是用 8 bit 实现针对某一个字符的标识，如美国信息交换标准代码（American standard code for information interchange，ASCII）。但由于 8 bit 只能提供 256 个编码定义，可用于编码的值太少，因此无法表示汉字。针对汉字，利用双字节（两个 8 bit，可支持 65 536 个汉字）实现编码。

可利用 Chrome 浏览器观察各种编码的效果。但 Chrome 浏览器在其编号为 55 的版本后移除了网站设置编码的功能，因此需添加插件 Set Character Encoding，利用该插件可手动指定编码格式。默认编码为 UTF-8，这里改成 GBK 之后发现页面为乱码。

ASCII 是基于拉丁字母的一套编码系统，主要用于显示现代英语和常用符号，是现今最通用的单字节编码系统。它的编码标准为 ISO-8859-1。通俗来说，ASCII 适用于针对英文字母加上标点符号的场景。

英语是字母文字，其常用单词均可以利用 26 个字母拼接实现，因此 ASCII 编码可满足英语环境。但在面对形意文字时，使用 ASCII 编码会有非常大的问题。中文是典型的形意文字，常用的文字数量达到 3 500 个以上，仅仅利用 8 bit 提供的 256 个编码远远无法满足编码需求。

利用双字节字符集（double byte character set，DBCS）可很好地解决编码不足的问题。常用的双字节字符集包括 GB2312、GBK 和 GB18030 等中文编码，使两字节长的汉字字符和一字节长的英文字符并存。图 1-1 所示是服务器的响应包示例。其中，利用 Content-Type 中的 charset 标识网页的编码格式。

```
HTTP/1.1  200    OK
Server:   Apache-Coyote/1.1
Content-Type:  text/html; charset=UTF-8
Content-Length:   4240
Date: Tue,05  Apr  2016 07:42:32  GMT
Connection:   close
```

图 1-1　HTTP 响应包中定义了当前页面的编码格式

不同国家和地区采用的编码不一致，因此无法正常显示所有字符的情况时有发生，即出现了乱码的情况。Unicode 编码主要解决多种语言环境下的统一集合问题，为各种语言中的每一个字符设定了统一且唯一的数字编号，以满足跨语言、跨平台进行文本转换、处理的要求。用来给 Unicode 字符集编码的标准有很多种，如 UTF-8、UTF-7、UTF-16、UnicodeLittle、UnicodeBig 等。

在国内，早期的站点大多使用 GBK 的编码方式实现中文显示，但目前主流站点都基于 UTF-8 进行中文显示。这主要是因为 UTF-8 支持多种语言环境，所以在多语言环境下使用 UTF-8 编码可大大降低客户端乱码的可能性。但需注意的是，UTF-8 是三字节编码，GBK 是双字节编码。因此在对大量内容编码时，UTF-8 编码所需的存储空间会多于 GBK。目前存储空间及网络带宽充足，因此在实际使用中二者没有明显的区别。

1.4.2 传输过程的编码

在 HTML 中，利用“/”“?”“&”等符号实现针对特定字符的内容定义，如规定访问路径、参数名称及间隔等。如果正常提交的参数里出现这类字符，势必会对正常 URL 解析造成影响。因此传输过程中进行编码的目的就是解决这个问题。常见编码有以下几种。

1. URL 编码

RFC 3986 文档规定，URL 中只允许包含英文字母（a ~ z、A ~ Z）、数字（0 ~ 9）、

4 个特殊字符（-、_、.、 ~ ）以及所有保留字符。在实际 Web 应用中，所使用的字符不止这些，如用户输入参数中还带有单引号、百分号、中文等。因此，需要对 URL 中的非允许字符进行编码。

URL 编码主体采用的是 ASCII 编码表，编码方式是用 %（百分号）加上两位字符代表一个字节。例如，单引号在 ASCII 中的十六进制编码为 27，在 URL 编码中就是 %27。对于中文字符，会先确认当前页面所用的编码格式。若当前页面使用 UTF-8 编码，则会先将中文字符转换成 UTF-8 编码，然后在每个字符的每一组编码前添加 %，这样就完成了 URL 编码。下面是一个实例。

URL 编码前为：

HTTP://172.29.152.23/loginPage.jsp?name= 测试 &passwd=ww121%$

URL 编码后为：

HTTP://172.29.152.23/loginPage.jsp?name=%E6%B5%8B%E8%AF%95&passwd=ww121%25$

假设当前页面为 UTF-8 编码。可以看到，URL 编码里针对参数“ww121%$”中的“%”进行了编码，编码结果为“%25”。针对中文字符“测试”，URL 编码为“%E6%B5%8B%E8%AF%95”。再查询“测试”字符的 UTF-8 编码，其十六进制编码就是“E6B58BE8AF95”，见表 1-1。

表 1-1　“测试”字符 UTF-8 编码

字　符	编码十进制	编码十六进制
测	15119755	E6B58B
试	15249301	E8AF95

以上过程很好地演示了 URL 编码针对中文字符的编码方式。了解 URL 编码方式会对后续攻防技术的学习有非常大的帮助。

2. Base64 编码

Base64 编码方式的核心逻辑是将 3 个字节的二进制数据（共 24 位）划分为 4 个 6 位的单元，每个单元对应一个 0~63 的十进制数值，再映射到由大小写英文字母、数字及“+”“/”组成的 64 个字符集，若二进制数据长度不是 4 的整数倍，则通过在末尾补“=”的方式完成填充，以此实现二进制数据向文本格式的转换。下面给出两个例子。

（1）编码前：Base64 编码；编码后：YmFzZTY057yW56CB（16 位）。

（2）编码前：Base64 编码 1 测试；编码后：YmFzZTY057yW56CBMea1i+ivlQ==。

Base64 编码非常好识别，含有大小写字母及 +、-、= 等符号，各种在线解码工具均可对 Base64 编码进行解码。利用 Burp Suite 的 Decoder 模块也可实现此类功能，如图 1-2 所示。

图 1-2　Burp Suite 的 Decoder 模块

Base64 编码可用于在 HTTP 环境下传递较长的标识信息，最早被用于邮件的传输。例如，在 HTTP Basic 认证中利用 Base64 对用户密码编码后进行传输。

3. HTML 字符实体

HTML 字符实体（character entity）是用来表示 HTML 中危险字符的方案，也是解决跨站脚本攻击的有效手段。

以常见的跨站脚本代码为例：<script>alert(/xss/)</script>。在这段代码由客户端提交到 Web 页面后，由于语句中含有 <script> 标签，会导致 HTML 页面将其当作 JavaScript 代码来执行。因此，在 HTML 内容中不能使用小于号（<）和大于号（>），否则浏览器会误认为它们是标签。

在日常应用中，如果需要正确地显示危险字符，可以使用 HTML 字符实体来实现。HTML 字符实体的特点是以 & 开头，并以分号结尾。例如，“<”的编码是“<”。在上述语句中，当用户提交的参数为“<script>alert(/xss/)</script>”时，经过 HTML 字符实体处理后，可得到“< script> alert(/xss/)< /script>”。这样就解决了危险字符的显示问题。

类似的编码类型还有很多，并且适用场景有所不同。编码的初衷是解决不同类型组件传递信息的一致性。但随着攻防技术的发展，编码也会根据其自身特点产生各类安全隐患。例如，之前常见的 GBK 编码，在 SQL 注入、XSS 环境下都存在宽字节的安全隐患。

1.4.3　Web 系统中的加密方法

标准的加密方法是对用户提交的参数（如密码、特定内容等）进行加密后再传输，避免参数在传输过程中被劫持，导致用户数据丢失。当数据传输到 Web 服务器时，将参数解密后处理。这个过程中存在两种情况。

1. 不需要服务器知道明文的内容

这种情况常见于用户的隐私信息，如用户密码。Web 系统在存储用户密码时不会直

接存储密码明文，而是预先设定加密算法，将用户的隐私信息加密后存储在数据库中。这样可以在系统运维过程中避免管理人员直接观察并获取用户的密码信息。这种情况下，经常利用 MD5/SHA-1 实现加密。

严格来说，MD5/SHA-1 是一种信息摘要算法，可将任意长度的明文内容转换成长度固定的密文，并且针对信息摘要的过程不可逆，但针对相同内容每次执行算法得到的密文完全相同。Web 系统存储的内容就是经过 MD5/SHA-1 转换后的密文。用户在客户端利用 MD5/SHA-1 将转换后的密文传输到 Web 系统，Web 系统再将用户密文与数据库中的密文进行比对即可。直接使用 MD5/SHA-1 并不安全，毕竟有大量彩虹表（存储明文与密文的表）存在，可间接达到密码破解的效果。因此，Web 系统常用 SALT 方式提升破解难度，但过于简单的 SALT 也会存在一定安全隐患，如图 1-3 所示。

图 1-3　支持添加 SALT 的 MD5 反向查询网站

除此之外，MD5/SHA-1 还存在碰撞问题，结果是不同明文利用 MD5 或 SHA-1 计算之后得到的密文完全相同，这个问题带来的影响远比彩虹表的威胁更大。考虑到安全情况，推荐使用 SHA-256/512 来提升安全性。但从性能角度考虑，MD5 或 SHA-1 的处理速度明显优于 SHA-256/512。因此在实际业务中需要综合业务系统安全需求及实际情况选择使用。

2. 需要服务器知道明文的内容

除用户的个人隐私信息之外，客户端发起的请求中还包含大量需要服务器处理的内容，如订单信息、留言等。由于 HTTP 在传输过程中并不会对其中的内容加密，就会导致在传输过程中内容被抓包，所以在传输过程中，加密的最大意义还是避免内容泄露，利用 HTTPS 可以有效解决这些问题。国内大量云厂商也支持采用这种方式进行安全的连接，例如，腾讯云提供了 SSL 证书功能，如图 1-4 所示。网站可根据自身需求选择对应的服务器。

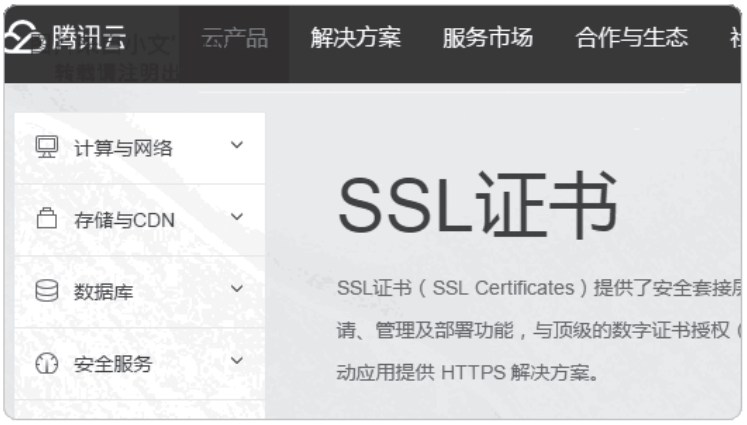

图 1-4　腾讯云提供 SSL 证书

需要注意的是，HTTPS 并不是完全免费的服务，受制于成本问题，多数大型站点仍然会采用 HTTP 开展业务。那么，要在 HTTP 下保障传输安全，可利用对称加密措施，如高级加密标准（AES）方式等。需要注意的是，由于 Web 站点始终在用户浏览器上，那么相对应的加密算法也处于公开状态，所以，针对这种情况，考虑更多的应该是加密算法的单次持续时间及重复程度。当然，也可以利用 JavaScript 混淆技术来提升加密算法的安全性。但最有效的手段是优化整体业务流程，从根本上降低需要加密传参的业务数量。

课后实践

1. 列举 Web 3.0 的优缺点。
2. Web 安全的核心问题有哪些？
3. 简述 HTTPS 的安全性特点。