



安全技术经典译丛

SQL注入攻击 与防御 (第2版)

SQL Injection Attacks and Defense, Second Edition

[美] Justin Clarke 著

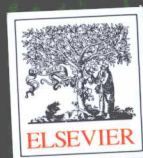
施宏斌 叶慷 译

上一版荣获2009 Bejtlich最佳图书奖

理解、发现、利用和防御SQL注入攻击的最佳参考

融入最新的SQL注入研究成果，涵盖丰富的示例和详尽分析

适用于Oracle、SQL Server、MySQL、PostgreSQL等数据库平台



清华大学出版社

安全技术经典译丛

SQL 注入攻击与防御

(第 2 版)

[美] Justin Clarke 著
施宏斌 叶隰 译

清华大学出版社

北 京

SQL Injection Attacks and Defense, Second Edition

Justin Clarke

ISBN: 978-1-59749-963-7

Copyright © 2012 by Elsevier. All rights reserved.

Authorized Simplified Chinese translation edition published by the Proprietor.

Copyright © 2013 by Elsevier (Singapore) Pte Ltd.

All rights reserved.

Published in China by Tsinghua University Press under special arrangement with Elsevier (Singapore) Pte Ltd.. This edition is authorized for sale in China only, excluding Hong Kong, Macau SAR and Taiwan. Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书简体中文版由 Elsevier (Singapore) Pte Ltd. 授予清华大学出版社在中国大陆地区(不包括香港、澳门特别行政区以及台湾地区)出版与发行。未经许可之出口, 视为违反著作权法, 将受法律之制裁。

北京市版权局著作权合同登记号: 01-2013-2564

本书封面贴有 Elsevier 防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

SQL 注入攻击与防御: 第 2 版/(美)克拉克(Clarke, J.) 著; 施宏斌, 叶慷 译.

—北京: 清华大学出版社, 2013.10

(安全技术经典译丛)

书名原文: SQL Injection Attacks and Defense, Second Edition

ISBN 978-7-302-34005-8

I. ①S… II. ①克… ②施… ③叶… III. ①关系数据库系统 IV. ①TP311.138

中国版本图书馆 CIP 数据核字(2013)第 228789 号

责任编辑: 王 军 李维杰

装帧设计: 牛静敏

责任校对: 成凤进

责任印制: 李红英

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 邮 编: 100084

社 总 机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者: 清华大学印刷厂

装 订 者: 北京市密云县京文制本装订厂

经 销: 全国新华书店

开 本: 185mm×260mm 印 张: 27.5 字 数: 704 千字

版 次: 2010 年 6 月第 1 版 2013 年 10 月第 2 版 印 次: 2013 年 10 月第 1 次印刷

印 数: 1~4000

定 价: 59.80 元

作者简介

Justin Clarke 是 Gotham Digital Science 公司的共同创办人和总监，Gotham Digital Science 是一家安全顾问公司，为客户提供识别、预防和管理安全风险的服务。在网络安全测试和软件领域，他有 15 年以上的工作经验。他还为美国、英国和新西兰等地的大型金融、零售和技术客户提供软件服务。

Justin 是很多计算机安全书籍的特约撰稿人，也是很多安全会议的演讲嘉宾和项目研究者，包括 Black Hat、EuSecWest、OSCON、ISACA、RSA、SANS、OWASP 和 British Computer Society。他是开源的 SQL 盲注漏洞利用工具 SQLBrute 的作者，还是 OWASP 在伦敦地区的负责人。

Justin 具有新西兰 Canterbury 大学计算机科学学士学位，还具有战略人力资源管理及会计 (Strategic Human Resources Management and Accounting) 专业的研究生文凭。或许这些学位对他都很有用。

编写者简介

Rodrigo Marcos Alvarez(CREST 顾问、MSc、BSc、CISSP、CNNA、OPST、MCP)是业界领先的渗透测试公司 SECFORCE 的技术总监。在不领导技术团队时, Rodrigo 还喜欢积极地参与安全评估的交付工作, 并亲自动手编写工具。他还致力于研究新的黑客技术。

Rodrigo 是 OWASP 项目的投稿人, 也是一名安全技术的研究者。他对通过模糊测试(fuzzing test)分析网络协议特别感兴趣。在其他项目中, 他发布了一款协议无关的 GUI 模糊器 TAOF, 还有一款可以在运行时对 TCP/UDP 代理的网络流量进行模糊处理的工具 proxyfuzz。Rodrigo 还在 Web 安全领域做出了不少贡献, 他发布了 bsishell——一个交互式 SQL 盲注的 Python shell, 此外他还开发了 TCP socket 重用攻击技术。

Kevvie Fowler(GCFA Gold、CISSP、MCTS、MCDBA、MCSD、MCSE)领导着 TELUS 安全智能分析项目, 他提出了高级事件分析和主动智能技术, 以保护客户免遭现在和新兴的安全威胁。

他还是安全研究和取证服务公司 Ringzero 的创建者和首席顾问。Kevvie 最近致力于数据库取证、rootkit 和原生加密缺陷等方面的研究。他出席了很多行业会议, 包括 Black Hat、SecTor 和 OWASP AppSec Asia。

Kevvie 是 *SQL Server Forensic Analysis* 一书的作者, 他还是多本信息安全和取证技术书籍的合著者。作为公认的 SANS 取证专家和 GIAC Advisory Board 成员, 他指导了对新兴安全和取证技术的研究。无论对于公共还是私人团体, Kevvie 都是一名值得信赖的顾问。他的领导才能已经在信息安全杂志 *Dark Reading and Kaspersky Threatpost* 中起到了重要作用。

Dave Hartley 是 MWR InfoSecurity 的首席安全顾问, 他是 CHECK 和 CREST 认证顾问(应用程序和基础设施)。MWR InfoSecurity 为客户提供识别、管理和降低信息安全风险方面的服务。

Dave 已经执行过大量的安全评估工作, 他还为很多不同领域的客户提供大量的顾问服务, 包括金融机构、企业、媒体、电信行业, 以及软件开发公司和世界范围内的政府组织。

Dave 还属于 CREST 和 NBISE 的技术顾问团, 他监督考试并合作开发新的 CREST 考试模块。CREST 是一个基于标准的组织机构, 它从事渗透测试工作, 并为个体顾问提供最佳实践的技术认证。Dave 还与 NBISE 联合, 为创建 US 考试中心而奔忙。

自从 1998 年以来, Dave 就一直在 IT 业界工作, 他的工作范围包括大量 IT 安全领域和学科。Dave 还是一位多产的作家, 他是多个信息安全期刊的主力作家。另外, 他还是 SQL 注入漏洞利用工具 Bobcat 的作者。

Alexander Kornbrust 专注于数据库安全的 Red-Database-Security 公司的创始人。他为世界范围内的客户提供数据库安全审计、安全培训和顾问服务。Alexander 还参与了业界领先的数据库安全工具 McAfee Security Scanner for Databases 的设计和开发。Alexander 自 1992 年起就从事与 Oracle 产品相关的工作, 他的专长是 Oracle 数据库和体系结构的安全问题。他已经向 Oracle 报告了 1200 多个安全 bug。他具有 Passau 大学计算机科学硕士学位(Diplom-Informatiker)。

Erlend Oftedal 是挪威奥斯陆 Bekk Consulting AS 公司的顾问, 多年以来一直领导 Bekk 的安全能力组(security competency group)。作为安全顾问和 Bekk 客户端的开发人员, 他花费了不少时间。他还负责代码审查和安全测试工作。

他针对 Web 应用程序安全进行了很多演讲, 包括在 Javazone 和 OWASP AppSec Europe 等软件开发和安全会议、用户组, 以及为挪威及海外大学所做的演讲。他是安全研究员, 在挪威的 OWASP 中具有重要作用。他还是 Norwegian HoneyNet 项目的成员。Erlend 具有挪威科技大学(NTNU)计算机科学硕士学位。

Gary O'Leary-Steele(CREST 顾问)是英国 Sec-1 有限公司的技术总监。目前, 他为各种客户提供高级渗透测试和安全顾问服务, 包括大量的大小在线零售商和金融领域的组织机构。他的专长包括 Web 应用程序安全评估、网络渗透测试和漏洞搜索。Gary 还是 Sec-1 Certified Network Security Professional(CNSP)培训课程的培训师和主要作者, 自从该培训开始以来, 已经有超过 3000 名学员, Gary 受到了 Microsoft、RSA、GFI、Splunk、IBM 和 Marshal Software 等公司的赞扬, Gary 在这些公司的商业应用程序中发现了一些安全缺陷。

Alberto Revelli 是安全研究员, 也是 Sqlninja 的作者。Sqlninja 是一款开源工具, 对基于 Microsoft SQL Server 的 Web 应用程序, 它已经成为利用 SQL 注入漏洞的必备武器。他的日常工作是为一家大型商品贸易公司服务, 对于引起他兴趣的任何东西, Alberto Revelli 都喜欢深入研究。

在 Alberto Revelli 的职业生涯中, 对大量公司进行了评估, 包括大型金融机构、电信企业、传媒公司和制造企业。很多安全会议都邀请他去演讲, 包括 EuSecWest、SOURCE、RSA、CONFidence、Shakacon 和 AthCon。

Alberto Revelli 居住在伦敦, 与他的女朋友一起享受着伦敦糟糕的天气和疯狂的夜生活。

Sumit "sid" Siddharth 领导着英国 7Safe 有限公司的渗透测试工作。他精通应用程序和数据库安全技术, 具有 6 年以上的渗透测试经验。Sid 创作了很多白皮书和工具。他是很多安全会议的演讲者或培训师, 包括 Black Hat、DEFCON、Troopers、OWASP Appsec 和 Sec-T 等。他还维护着著名的 IT 安全博客: www.notesosecure.com。

Marco Slaviero 是 SensePost 的合伙人, 他是 SensePost 实验室的带头人。在安全业界的 BlackHat USA 和 DefCon 等会议上, 他对包括 SQL 注入在内的多种安全主题发表过演讲。Marco 的专业领域包括与网络有关的应用程序测试、为 4 大洲的客户的高级顾问服务。

Marco 与他美丽的妻子 Juliette 生活在一起。

几年前 Marco 获得了 Pretoria 大学的硕士学位, 但这都是过去取得的成绩, 他对此毫不在意。

Dafydd Stuttard 是一位独立的安全顾问、作者和软件开发人员, 他精通 Web 应用程序的渗透测试和汇编软件。Dafydd 是畅销书 *Web Application Hacker's Handbook* 的作者。他的别名是 PortSwigger, 他创建了流行的 Web 应用程序黑客工具 Burp Suite。Dafydd 为安全会议以及在这个世界上其他地点举办的会议开发并提供培训课程。Dafydd 具有牛津大学哲学硕士和博士学位。

致 谢

首先再次感谢 Syngress 的编辑团队(特别感谢 Chris Katsaropoulos 和 Heather Scherer),感谢他们愿意再次出版一本由多名作者共同编写的图书。作为本书的主要撰稿人,还要感谢本书的作者团队,感谢他们齐心协力共同完成了本书。

前 言

自从 2009 年本书第 1 版出版以来又过去了不少时间,大约经过了 3 年之后,本书的第 2 版也已经面世。当我们在第 1 版中讨论 SQL 注入的理念时,SQL 注入已经出现了 10 多年,并且在本质上并没有新的改变。截至 2008 年(大约在发现 SQL 注入这一问题 10 年之后,当本书第 1 版刚开始成形时),对于什么是 SQL 注入、如何发现 SQL 注入漏洞,以及如何利用漏洞,人们依然没有综合性的理解,更不用说如何防御 SQL 注入漏洞,以及如何从一开始就避免 SQL 注入漏洞的出现。另外,普通的观点是 SQL 注入仅仅与 Web 应用程序有关,对于混合攻击或者作为一种渗透组织机构外部安全控制的方法而言,SQL 注入并不是一种危险因素——事实充分证明这种观点是错误的,在本书第 1 版付梓前后发生的黑客安全事件就是最好的说明(比如 Heartland Payment Systems 的安全事件)。

现在是 2012 年,笔者完成了本书的第 2 版,虽然在 SQL 注入的基础理论上只有很小的变化,但是 SQL 注入的技术已经不断进步,在将 SQL 注入应用于较新的领域方面已经有了新的发展,比如将 SQL 注入应用于移动应用程序,以及通过 HTML5 实现客户端 SQL 注入。另外,此书第 2 版还为我与本书的合著者提供了一次机会,对读者在第 1 版中提出的问题提供反馈。在第 2 版中,不但全面更新了本书的所有内容,还介绍了一些新的技术和方法。另外在第 2 版中还扩大了数据库的范围,包含对 PostgreSQL 数据库的介绍。在本书的各个章节中,都将 Microsoft SQL Server、Oracle、MySQL 和 PostgreSQL 数据库作为主要的数据库平台,并在相关内容中使用 Java、.NET 和 PHP 编写了代码示例。

本书总体上分为 4 个部分——理解 SQL 注入(第 1 章)、发现 SQL 注入(第 2~3 章)、利用 SQL 注入漏洞(第 4~7 章),以及防御 SQL 注入(第 8 章~10 章)。每一部分都有意针对不同的读者,从所有读者(理解 SQL 注入)、安全专家和渗透测试人员(发现和利用 SQL 注入漏洞),到管理数据库的开发专家和 IT 专家(发现和防御 SQL 注入)。为了使本书的内容更丰富,包含了第 11 章以提供参考资料,该章还包含了本书并未详细介绍的其他数据库平台。如果偶然遇到这样的平台,读者可以参考本书前面章节中讨论的各种技术。

下面是每一章的内容提要:

第 1 章——介绍什么是 SQL 注入,以及 SQL 注入是如何发生的。

第 2 章——介绍如何从 Web 应用程序前端发现 SQL 注入,包括如何检测可能存在的 SQL 注入漏洞、如何确认 SQL 注入漏洞的存在,以及如何自动发现 SQL 注入漏洞。

第 3 章——如何通过审查代码来发现 SQL 注入漏洞,既可以手工方式审查代码,也可以通过自动方式审查代码。

第 4 章——如何利用 SQL 注入漏洞,包括几种常见技术,比如 UNION 语句和条件语句、枚举数据库模式、盗取密码哈希,以及以自动化利用 SQL 注入漏洞。

第 5 章——如何利用 SQL 盲注漏洞,包括使用基于时间、基于响应和非主流通道返回数据。

第 6 章——介绍如何通过 SQL 注入利用操作系统的漏洞，包括读取和写入文件，以及通过 SQL 注入漏洞执行操作系统命令。

第 7 章——介绍利用漏洞的高级主题，包括如何利用二阶 SQL 注入漏洞、如何利用客户端 SQL 注入漏洞，以及如何通过 SQL 注入执行混合攻击。

第 8 章——介绍针对 SQL 注入的代码层防御，包括基于设计的方法、使用参数化查询、编码技术以及验证有效 SQL 注入方法。

第 9 章——介绍针对 SQL 注入在应用程序平台层次上的防御措施，包括使用运行时保护、数据库加固，以及如何减小 SQL 注入影响的安全部署方面的考虑。

第 10 章——介绍如何确认 SQL 注入攻击并从攻击中恢复，包括如何确定捕获 SQL 注入失败、确定 SQL 注入是否已经成功，以及在受到 SQL 注入攻击时如何对数据库进行恢复。

第 11 章——介绍 SQL 基础知识，为 Microsoft SQL Server、Oracle、MySQL 和 PostgreSQL 数据库平台的 SQL 注入提供快速参考，还包括在其他平台上执行 SQL 注入的细节，比如 DB2、Sybase、Access 和其他数据库。

目 录

第 1 章 什么是 SQL 注入.....1	
1.1 概述.....1	
1.2 理解 Web 应用的工作原理.....2	
1.2.1 一种简单的应用架构.....3	
1.2.2 一种较复杂的架构.....4	
1.3 理解 SQL 注入.....5	
1.4 理解 SQL 注入的产生过程.....10	
1.4.1 构造动态字符串.....10	
1.4.2 不安全的数据库配置.....16	
1.5 本章小结.....18	
1.6 快速解决方案.....18	
1.7 常见问题解答.....19	
第 2 章 SQL 注入测试.....21	
2.1 概述.....21	
2.2 寻找 SQL 注入.....21	
2.2.1 借助推理进行测试.....22	
2.2.2 数据库错误.....28	
2.2.3 应用程序的响应.....39	
2.2.4 SQL 盲注.....42	
2.3 确认 SQL 注入.....45	
2.3.1 区分数字和字符串.....46	
2.3.2 内联 SQL 注入.....46	
2.3.3 终止式 SQL 注入.....52	
2.3.4 时间延迟.....59	
2.4 自动寻找 SQL 注入.....60	
2.5 本章小结.....68	
2.6 快速解决方案.....68	
2.7 常见问题解答.....69	
第 3 章 复查代码中的 SQL 注入.....71	
3.1 概述.....71	
3.2 复查源代码中的 SQL 注入.....71	
3.2.1 危险的编码行为.....73	
3.2.2 危险的函数.....78	
3.2.3 跟踪数据.....81	
3.2.4 复查 Android 应用程序代码.....88	
3.2.5 复查 PL/SQL 和 T-SQL 代码.....93	
3.3 自动复查源代码.....99	
3.3.1 Graidit.....100	
3.3.2 YASCA.....101	
3.3.3 Pixy.....101	
3.3.4 AppCodeScan.....102	
3.3.5 OWASP LAPSE+项目.....102	
3.3.6 Microsoft SQL 注入源代码 分析器.....103	
3.3.7 CAT.NET.....103	
3.3.8 RIPS——PHP 脚本漏洞 的静态源代码分析器.....103	
3.3.9 CodePro AnalytiX.....104	
3.3.10 Teachable Static Analysis Workbench.....104	
3.3.11 商业源代码复查工具.....104	
3.3.12 Fortify 源代码分析器.....105	
3.3.13 Rational AppScan Source Edition.....106	
3.3.14 CodeSecure.....106	
3.3.15 Klocwork Solo.....106	
3.4 本章小结.....107	
3.5 快速解决方案.....107	
3.6 常见问题解答.....108	
第 4 章 利用 SQL 注入.....111	
4.1 概述.....111	
4.2 理解常见的漏洞利用技术.....112	
4.2.1 使用堆叠查询.....113	
4.2.2 在 Web 应用程序中利用 Oracle 漏洞.....114	
4.3 识别数据库.....114	
4.3.1 非盲跟踪.....115	
4.3.2 盲跟踪.....119	

4.4	使用 UNION 语句提取数据	120	4.11.3	BSQL	181
4.4.1	匹配列	121	4.11.4	其他工具	183
4.4.2	匹配数据类型	122	4.12	本章小结	183
4.5	使用条件语句	126	4.13	快速解决方案	184
4.5.1	方法 1: 基于时间	126	4.14	常见问题解答	185
4.5.2	方法 2: 基于错误	130			
4.5.3	方法 3: 基于内容	131	第 5 章	SQL 盲注利用	187
4.5.4	处理字符串	131	5.1	概述	187
4.5.5	扩展攻击	133	5.2	寻找并确认 SQL 盲注	188
4.5.6	利用 SQL 注入错误	134	5.2.1	强制产生通用错误	188
4.5.7	Oracle 中的错误消息	135	5.2.2	注入带副作用的查询	188
4.6	枚举数据库模式	139	5.2.3	拆分与平衡	189
4.6.1	SQL Server	139	5.2.4	常见的 SQL 盲注场景	191
4.6.2	MySQL	143	5.2.5	SQL 盲注技术	191
4.6.3	PostgreSQL	146	5.3	使用基于时间的技术	200
4.6.4	Oracle	147	5.3.1	延迟数据库查询	200
4.7	在 INSERT 查询中实施注入攻击	150	5.3.2	基于时间的推断应考虑的问题	207
4.7.1	第一种情形: 插入用户规定的数 据	150	5.4	使用基于响应的技术	207
4.7.2	第二种情形: 生成 INSERT 错误	153	5.4.1	MySQL 响应技术	208
4.7.3	其他情形	155	5.4.2	PostgreSQL 响应技术	209
4.8	提升权限	155	5.4.3	SQL Server 响应技术	210
4.8.1	SQL Server	155	5.4.4	Oracle 响应技术	211
4.8.2	在未打补丁的服务器上提升 权限	160	5.4.5	返回多位信息	213
4.8.3	SYS.DBMS_CDC_PUBLISH	162	5.5	使用非主流通道	214
4.9	窃取哈希口令	164	5.5.1	数据库连接	214
4.9.1	SQL Server	164	5.5.2	DNS 渗漏	216
4.9.2	MySQL	165	5.5.3	e-mail 渗漏	219
4.9.3	PostgreSQL	166	5.5.4	HTTP 渗漏	220
4.9.4	Oracle	166	5.5.5	ICMP 渗漏	222
4.10	带外通信	169	5.6	自动 SQL 盲注利用	222
4.10.1	e-mail	169	5.6.1	Absinthe	222
4.10.2	HTTP/DNS	172	5.6.2	BSQL Hacker	223
4.10.3	文件系统	173	5.6.3	SQLBrute	225
4.11	自动利用 SQL 注入	179	5.6.4	Sqlmap	226
4.11.1	Sqlmap	180	5.6.5	Sqlinja	227
4.11.2	Bobcat	181	5.6.6	Squeeza	228
			5.7	本章小结	229
			5.8	快速解决方案	229
			5.9	常见问题解答	231

第 6 章 利用操作系统	233	8.3 使用参数化语句	297
6.1 概述	233	8.3.1 Java 中的参数化语句	298
6.2 访问文件系统	234	8.3.2 .NET(C#)中的参数化 语句	299
6.2.1 读文件	234	8.3.3 PHP 中的参数化语句	300
6.2.2 写文件	248	8.3.4 PL/SQL 中的参数化语句	301
6.3 执行操作系统命令	257	8.4 移动应用中的参数化语句	302
6.4 巩固访问	269	8.4.1 iOS 应用程序中的参数化 语句	302
6.5 本章小结	271	8.4.2 Android 应用程序中的参数化 语句	302
6.6 快速解决方案	271	8.4.3 HTML5 浏览器存储中的参数 化语句	303
6.7 常见问题解答	272	8.5 输入验证	303
第 7 章 高级话题	273	8.5.1 白名单	303
7.1 概述	273	8.5.2 黑名单	306
7.2 避开输入过滤器	273	8.5.3 Java 中的输入验证	307
7.2.1 使用大小写变种	274	8.5.4 .NET 中的输入验证	308
7.2.2 使用 SQL 注释	274	8.5.5 PHP 中的输入验证	309
7.2.3 使用 URL 编码	275	8.5.6 在移动应用程序中检验 输入	309
7.2.4 使用动态查询执行	276	8.5.7 在 HTML5 中检验输入	309
7.2.5 使用空字节	278	8.6 编码输出	310
7.2.6 嵌套剥离后的表达式	278	8.7 规范化	316
7.2.7 利用截断	278	8.8 通过设计来避免 SQL 注入 的危险	319
7.2.8 避开自定义过滤器	280	8.8.1 使用存储过程	319
7.2.9 使用非标准入口点	280	8.8.2 使用抽象层	320
7.3 利用二阶 SQL 注入	282	8.8.3 处理敏感数据	321
7.4 客户端 SQL 注入漏洞	285	8.8.4 避免明显的对象名	322
7.4.1 访问本地数据库	286	8.8.5 创建 honeypot	322
7.4.2 攻击客户端数据库	286	8.8.6 附加的安全开发资源	323
7.5 使用混合攻击	288	8.9 本章小结	324
7.5.1 利用捕获的数据	288	8.10 快速解决方案	324
7.5.2 创建跨站脚本	288	8.11 常见问题解答	325
7.5.3 在 Oracle 上运行操作系统 命令	289	第 9 章 平台层防御	327
7.5.4 利用验证过的漏洞	289	9.1 概述	327
7.6 本章小结	290	9.2 使用运行时保护	327
7.7 快速解决方案	291	9.2.1 Web 应用防火墙	328
7.8 常见问题解答	292		
第 8 章 代码层防御	293		
8.1 概述	293		
8.2 领域驱动的安全	293		

9.2.2	截断过滤器	333	11.2	SQL 入门	387
9.2.3	不可编辑与可编辑的输入 保护	336	11.3	SQL 注入快速参考	393
9.2.4	URL 策略与页面层策略	337	11.3.1	识别 SQL 注入漏洞	393
9.2.5	面向方面编程	338	11.3.2	识别数据库平台	395
9.2.6	应用程序入侵检测系统	338	11.3.3	Microsoft SQL Server 备忘单	398
9.2.7	数据库防火墙	339	11.3.4	MySQL 备忘单	406
9.3	确保数据库安全	339	11.3.5	Oracle 备忘单	408
9.3.1	锁定应用程序数据	339	11.3.6	PostgreSQL 备忘单	413
9.3.2	锁定数据库服务器	343	11.4	避开输入验证过滤器	415
9.4	额外的部署考虑	345	11.4.1	引号过滤器	415
9.4.1	最小化不必要信息的泄露	345	11.4.2	HTTP 编码	416
9.4.2	提高 Web 服务器日志的详细 程度	349	11.5	排查 SQL 注入攻击	417
9.4.3	将 Web 服务器和数据库服务 器分别部署在独立主机上	349	11.6	其他平台上的 SQL 注入	419
9.4.4	配置网络访问控制	349	11.6.1	DB2 备忘单	420
9.5	本章小结	349	11.6.2	Informix 备忘单	420
9.6	快速解决方案	350	11.6.3	Ingres 备忘单	422
9.7	常见问题解答	350	11.6.4	Sybase 备忘单	423
			11.6.5	Microsoft Access	424
			11.7	资源	424
			11.7.1	SQL 注入白皮书	424
			11.7.2	SQL 注入备忘单	424
			11.7.3	SQL 注入利用工具	425
			11.7.4	口令破解工具	425
			11.8	快速解决方案	425
第 10 章	确认并从 SQL 注入攻击中 恢复	353			
10.1	简介	353			
10.2	调查可疑的 SQL 注入攻击	353			
10.2.1	取证的合理实践	354			
10.2.2	分析数字化证据	355			
10.3	如果你是受害者, 该 怎么办?	376			
10.3.1	遏制安全事件	376			
10.3.2	评估涉及的数据	377			
10.3.3	通知相应人员	377			
10.3.4	确定攻击者在系统上执行了 哪些操作?	378			
10.3.5	从 SQL 注入攻击中恢复	378			
10.4	小结	382			
10.5	快速解决方案	383			
10.6	常见问题解答	384			
第 11 章	参考资料	387			
11.1	概述	387			

第 1 章 什么是 SQL 注入

本章目标:

- 理解 Web 应用的工作原理
- 理解 SQL 注入
- 理解 SQL 注入的产生过程

1.1 概述

很多人声称自己了解 SQL 注入,但他们听说或经历的情况都是比较常见的。SQL 注入是影响企业运营且最具破坏性的漏洞之一,它会泄露保存在应用程序数据库中的敏感信息,包括用户名、口令、姓名、地址、电话号码以及信用卡明细等易被利用的信息。

那么,应该怎样来准确定义 SQL 注入呢?SQL 注入(SQL Injection)是这样一种漏洞:应用程序在向后台数据库传递 SQL(Structured Query Language, 结构化查询语言)查询时,如果为攻击者提供了影响该查询的能力,就会引发 SQL 注入。攻击者通过影响传递给数据库的内容来修改 SQL 自身的语法和功能,并且会影响 SQL 所支持数据库和操作系统的功能和灵活性。SQL 注入不只是一种会影响 Web 应用的漏洞;对于任何从不可信源获取输入的代码来说,如果使用该输入来构造动态 SQL 语句,就很可能也会受到攻击(例如,客户端/服务器架构中的“胖客户端”程序)。在过去,典型的 SQL 注入更多的是针对服务器端的数据库,然而根据目前的 HTML5 规范,攻击者可以采用完全相同的办法,执行 JavaScript 或其他代码访问客户端数据库以窃取数据。移动应用程序(比如 Android 平台)也与之类似,恶意应用程序或客户端脚本也可以采用类似的方式进行 SQL 注入攻击(请访问 labs.mwrinfosecurity.com/notices/webcontentresolver/ 以获取更详细的信息)。

自 SQL 数据库开始连接至 Web 应用起,SQL 注入就可能已经存在。Rain Forest Puppy 因首次发现它(或至少将其引入了公众的视野)而备受赞誉。1998 年圣诞节,Rain Forest Puppy 为 *Phrack*(www.phrack.com/issues.html?issue=54&id=8#article, 一本由黑客创办且面向黑客的电子杂志)撰写了一篇名为“NT Web Technology Vulnerabilities”(NT Web 技术漏洞)的文章。2000 年早期,Rain Forest Puppy 还发布了一篇关于 SQL 注入的报告(“How I hacked PacketStorm”, 位于 www.wiretrip.net/rfp/txt/rfp2k01.txt),其中详述了如何使用 SQL 注入来破坏一个当时很流行的 Web 站点。自此,许多研究人员开始研究并细化利用 SQL 注入进行攻击的技术。但时至今日,仍有许多开发人员和专家对 SQL 注入不甚了解。

本章将介绍 SQL 注入的成因。首先概述 Web 应用通用的构建方式,为理解 SQL 注入的产生过程提供一些背景知识。接下来从 Web 应用的代码层介绍引发 SQL 注入的因素以及哪些开

发实践和行为会引发 SQL 注入。

1.2 理解 Web 应用的工作原理

大多数人在日常生活中都会用到 Web 应用。有时是作为假期生活的一部分，有时是为了访问 e-mail、预定假期、从在线商店购买商品或是查看感兴趣的新闻消息等。Web 应用的形式有很多种。

不管是用何种语言编写的 Web 应用，有一点是相同的：它们都具有交互性并且多半是数据库驱动的。在互联网中，数据库驱动的 Web 应用非常普遍。它们通常都包含一个后台数据库和很多 Web 页面，这些页面中包含了使用某种编程语言编写的服务器端脚本，而这些脚本则能够根据 Web 页面与用户的交互从数据库中提取特定的信息。电子商务是数据库驱动的 Web 应用的最常见形式之一。电子商务应用的很多信息，如产品信息、库存水平、价格、邮资、包装成本等均保存在数据库中。如果读者曾经从电子零售商那里在线购买过商品和产品，那么应该不会对这种类型的应用感到陌生。数据库驱动的 Web 应用通常包含三层：表示层(Web 浏览器或呈现引擎)、逻辑层(如 C#、ASP、.NET、PHP、JSP 等编程语言)和存储层(如 Microsoft SQL Server、MySQL、Oracle 等数据库)。Web 浏览器(表示层，如 Internet Explorer、Safari、Firefox 等)向中间层(逻辑层)发送请求，中间层通过查询、更新数据库(存储层)来响应该请求。

下面看一个在线零售商店的例子。该在线商店提供了一个搜索表单，顾客可以按特定的兴趣对商品进行过滤、分类。另外，它还提供了对所显示商品作进一步筛选的选项，以满足顾客在经济上的预算需求。可以使用下列 URL 查看商店中所有价格低于\$100 的商品：<http://www.victim.com/products.php?val=100>。

下列 PHP 脚本说明了如何将用户输入(val)传递给动态创建的 SQL 语句。当请求上述 URL 时，将会执行下列 PHP 代码段：

```
// 连接到数据库
$conn = mysql_connect("localhost","username","password");

//使用输入动态创建 SQL 语句
$query = "SELECT * FROM Products WHERE Price < '$_GET['val']' " .
        "ORDER BY ProductDescription";

//对数据库执行查询
$result = mysql_query($query);

//迭代返回的记录集
while($row = mysql_fetch_array($result, MYSQL_ASSOC))
{
    //将结果显示在浏览器中

    echo "Description : {$row['ProductDescription']} <br>" .
        "Product ID : {$row['ProductID']} <br>" .
        "Price : {$row['Price']} <br><br>";
}
}
```

接下来的代码示例更清晰地说明了 PHP 脚本构造并执行的 SQL 语句。该语句返回数据库中所有价格低于\$100 的商品，之后在 Web 浏览器上显示并呈现这些商品以方便顾客在预算范

围内继续购物。一般来说，所有可交互的数据库驱动的 Web 应用均以相同的(至少是类似的)方式运行：

```
SELECT *
FROM Products
WHERE Price <'100.00'
ORDER BY ProductDescription;
```

1.2.1 一种简单的应用架构

前面讲过，数据库驱动的 Web 应用通常包含三层：表示层、逻辑层和存储层。为更好地帮助读者理解 Web 应用技术是如何进行交互的，从而为用户带来功能丰富的 Web 体验，我们借助图 1-1 来说明前面描述的那个简单的三层架构示例。

表示层是应用的最高层，它显示与商品浏览、购买、购物车等服务相关的信息，并通过将结果输出到浏览器/客户端层和网络上的所有其他层来与应用架构的其他层进行通信。逻辑层是从表示层剥离出来的，作为单独的一层，它通过执行细节处理来控制应用的功能。数据层包括数据库服务器，用于对信息进行存储和检索。数据层保证数据独立于应用服务器或业务逻辑。将数据作为单独的一层还可以提高程序的可扩展性和性能。在图 1-1 中，Web 浏览器(表示层)向中间层(逻辑层)发送请求，中间层通过查询、更新数据库(存储层)响应该请求。三层架构中一条最基本的规则是：表示层不应直接与数据层通信。在三层架构中，所有通信都必须经过中间层。从概念上看，三层架构是一种线性关系。

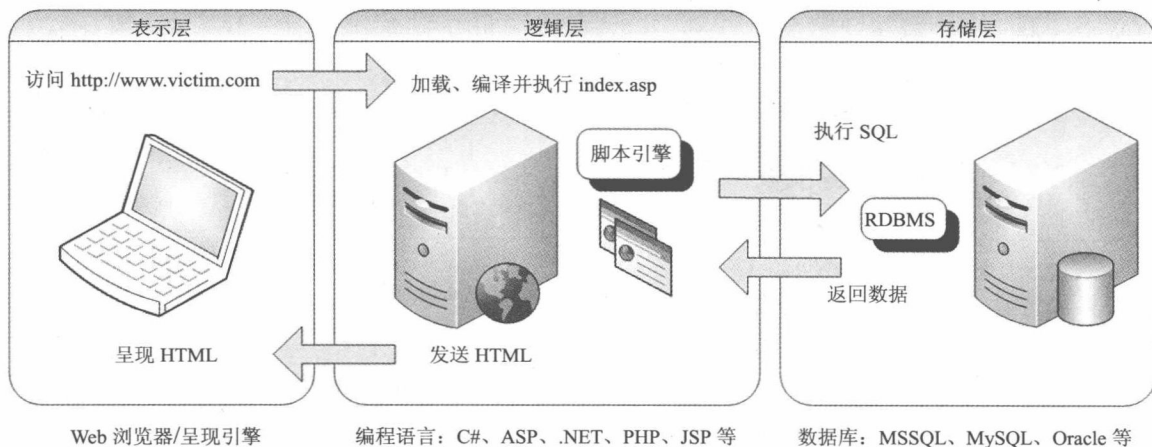


图 1-1 简单的三层架构

在图 1-1 中，用户激活 Web 浏览器并连接到 <http://www.victim.com>。位于逻辑层的 Web 服务器从文件系统中加载脚本并将其传递给脚本引擎，脚本引擎负责解析并执行脚本。脚本使用数据库连接程序打开存储层连接并对数据库执行 SQL 语句。数据库将数据返回给数据库连接程序，后者将其传递给逻辑层的脚本引擎。逻辑层在将 Web 页面以 HTML 格式返回给表示层的用户的 Web 浏览器之前，先执行相关的应用或业务逻辑规则。用户的 Web 浏览器呈现 HTML 并借助代码的图形化表示展现给用户。所有操作都将在数秒内完成，并且对用户是透明的。

1.2.2 一种较复杂的架构

三层解决方案不具有扩展性,所以最近几年研究人员不断地对三层架构进行改进,并在可扩展性和可维护性基础之上创建了一种新概念: n 层应用程序开发范式。其中包括一种 4 层解决方案,该方案在 Web 服务器和数据库之间使用了一层中间件(通常称为应用服务器)。 n 层架构中的应用服务器负责将 API(应用编程接口)提供给业务逻辑和业务流程以供程序使用。可以根据需要引入其他的 Web 服务器。此外,应用服务器可以与多个数据源通信,包括数据库、大型机以及其他旧式系统。

图 1-2 描绘了一种简单的 4 层架构。

在图 1-2 中,Web 浏览器(表示层)向中间层(逻辑层)发送请求,后者依次调用由位于应用层的应用服务器提供的 API,应用层通过查询、更新数据库(存储层)来响应该请求。

在图 1-2 中,用户激活 Web 浏览器并连接到 `http://www.victim.com`。位于逻辑层的 Web 服务器从文件系统中加载脚本并将其传递给脚本引擎,脚本引擎负责解析并执行脚本。脚本调用由位于应用层的应用服务器提供的 API。应用服务器使用数据库连接程序打开存储层连接并对数据库执行 SQL 语句。数据库将数据返回给数据库连接程序,应用服务器在将数据返回给 Web 服务器之前先执行相关的应用或业务逻辑规则。Web 服务器在将数据以 HTML 格式返回给表示层的用户的 Web 浏览器之前先执行最后的有关逻辑。用户的 Web 浏览器呈现 HTML 并借助代码的图形化表示展现给用户。所有操作都将在数秒内完成,并且对用户是透明的。

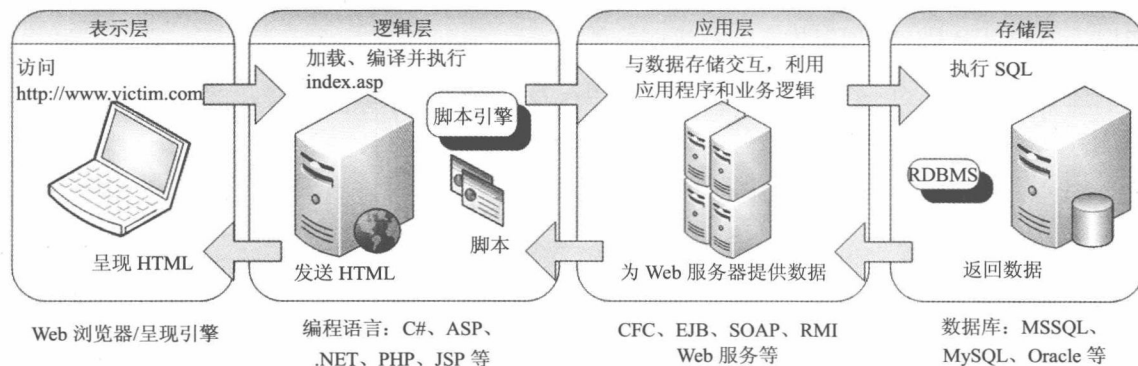


图 1-2 4 层架构

分层架构的基本思想是将应用分解成多个逻辑块(或层),其中每一层都分配有通用或特定的角色。各个层可以部署在不同的机器上,或者部署于同一台机器上,但实际在概念上是彼此分离的。使用的层越多,每一层的角色就越具体。将应用的职责分成多个层能使应用更易于扩展,可以更好地为开发人员分配开发任务,提高应用的可读性和组件的可复用性。该方法还可以通过消除单点失败来提高应用的健壮性。例如,决定更换数据库提供商时,只需修改应用层的相关部分即可,表示层和逻辑层可保持不变。在互联网上,3 层架构和 4 层架构是最常见的部署架构。正如前面介绍的, n 层架构非常灵活,在概念上它支持多层之间的逻辑分离,并且支持以多种方式进行部署。