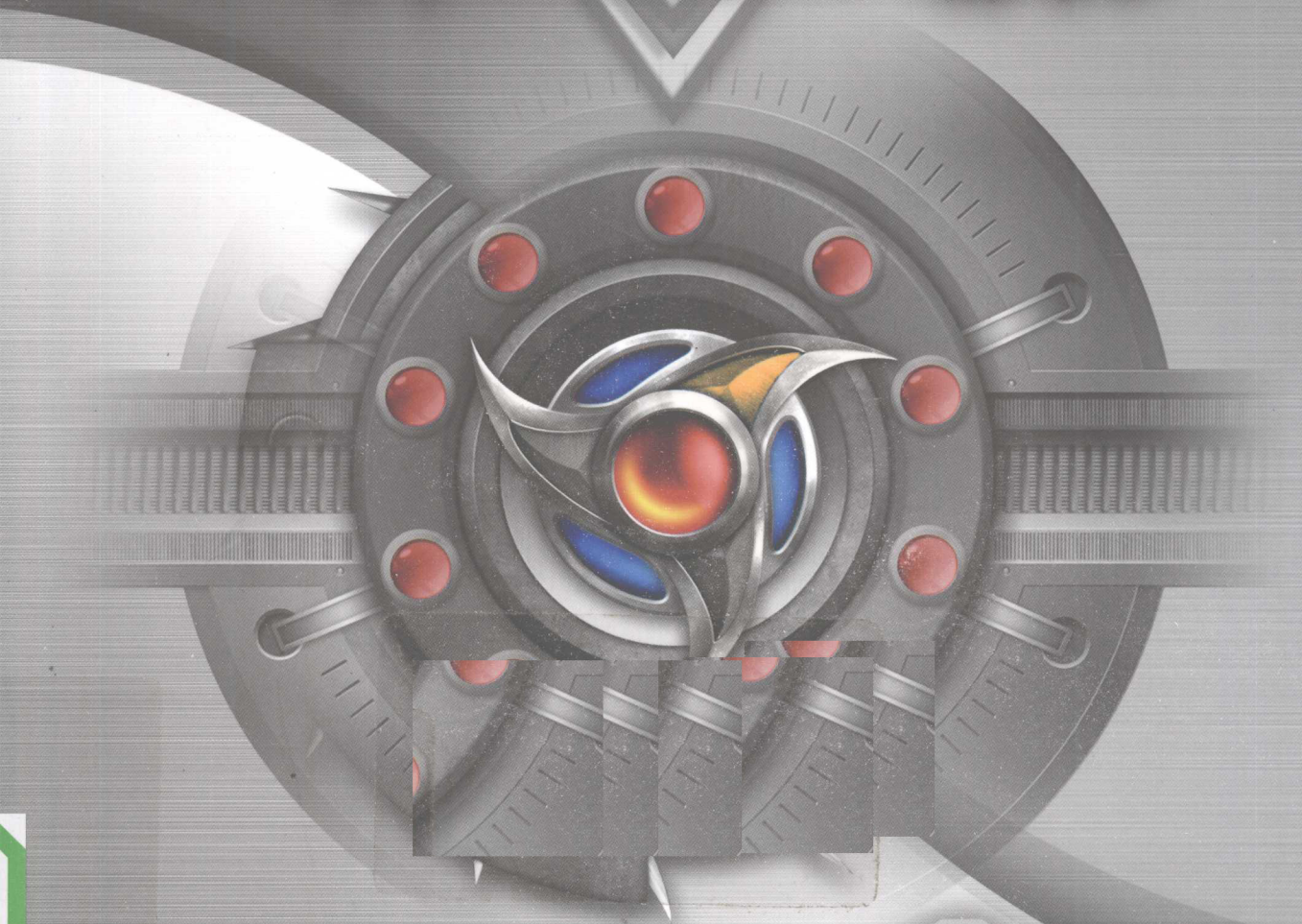
安全技术经典译丛

SQL Injection Attacks and Defense

SQL注入攻击与防御



(美) Justin Clarke 等著

黄晓磊 李 化 译

- 唯一一本关于SQL注入攻击与防御的专业书籍
- 理解、发现、利用和防御SQL注入的最佳指导
- 见解精辟，丰富、精彩的SQL注入示例及防御策略
- 作者多年长期实践经验的总结

清华大学出版社

清华大学出版社
北 京

SQL Injection Attacks and Defense
Justin Clarke, et al.
EISBN：978-1-597-49424-3

ISBN: 978-9812724564

图书在版编目(CIP)数据

SQL 注入攻击与防御/(美)克拉克(Clarke, J.) 等 著；黄晓磊，李化 译. —北京：清华大学出版社，2010.6
(安全技术经典译丛)
书名原文：SQL Injection Attacks and Defense

ISBN 978-7-302-22413-6

Ⅰ. S… Ⅱ. ①克… ②黄… ③李… Ⅲ. 关系数据库—数据库管理系统，SQL—安全技术 Ⅳ. TP311.138

中国版本图书馆 CIP 数据核字(2010)第 063869 号

责任编辑：王　军　李维杰
封面设计：久久度文化
版式设计：康　博
责任校对：胡雁翎
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**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：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：23　**字　数**：589 千字
版　　次：2010 年 6 月第 1 版　　**印　　次**：2010 年 6 月第1 次印刷
印　　数：1～4000
定　　价：48.00 元

产品编号：033608-01

译 者 序

十几年前，基于数据库的 Web 应用刚流行时，几乎所有开发商都忽略了 SQL 注入漏洞，导致当时大多数网站的登录入口形同虚设。时至今日，Web 应用已愈加成熟，安全性也不断得到加强。遗憾的是，针对 SQL 注入漏洞的各种攻击工具也在推陈出新，不断地向安全管理人员发出新的挑战。如何最大程度地降低 SQL 注入风险，从根本上实施 SQL 注入防御，成为网络管理人员和开发人员亟需解决的“烫手山芋”。

现在网络上关于 SQL 注入方面的教程比较零散，大多针对某一类具体应用，难以作为预防 SQL 注入的完整解决方案。本书弥补了这一缺憾！本书作者均是专门研究 SQL 注入的安全专家，他们集众家之长，对应用程序的基本编码和升级维护进行全程跟踪，详细讲解可能引发 SQL 注入的行为以及攻击者的利用要素，并结合长期实践经验提出了相应的解决方案。SQL 注入利用的是正常的 HTTP 服务端口，表面上和正常的 Web 访问没有差别，隐蔽性极强。针对这种情况，书中重点讲解了 SQL 注入的排查方法和可以借助的工具，总结了常见的利用 SQL 注入漏洞的方法。开发人员和系统管理人员在 SQL 注入防御中扮演着重要角色，因此，书中专门从代码层和系统层角度介绍了避免 SQL 注入的各种策略和需要考虑的问题。

全书共 10 章，分别介绍了 SQL 注入的基本概念，如何发现、确认并利用 SQL 注入和 SQL 盲注，利用操作系统防御 SQL 注入，SQL 注入的一些高级话题，代码层和平台层防御等知识，书中主要针对的是 Microsoft SQL Server、MySQL 和 Oracle 这三大主流数据库。本书注重于实践，涉及的内容也比较前沿，另外，还包含了大量翔实的案例，它们都具有很好的现实指导作用，读者可从中学到最新的攻击和防御技术。

本书主要由黄晓磊和李化翻译完成，全书由李化统稿。由于本书内容较新、知识面广且译者水平有限，译文中难免存在错误之处，敬请读者批评指正。

译者

目　录

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，则会引发 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，如果使用了该输入来构造动态 SQL 语句，那么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)。

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安全专家对 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：

```
// connect to the database
$conn = mysql_connect("localhost","username","password");

// dynamically build the sql statement with the input
$query = "SELECT * FROM Products WHERE Price < '$_GET["val"]' " .
         "ORDER BY ProductDescription";

// execute the query against the database
$result = mysql_query($query);

// iterate through the record set
while($row = mysql_fetch_array($result, MYSQL_ASSOC))
{
    // display the results to the browser

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

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

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。

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、购物车内容等服务相关的信息，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件层。从概念上看，三层架构是一种线性关系。

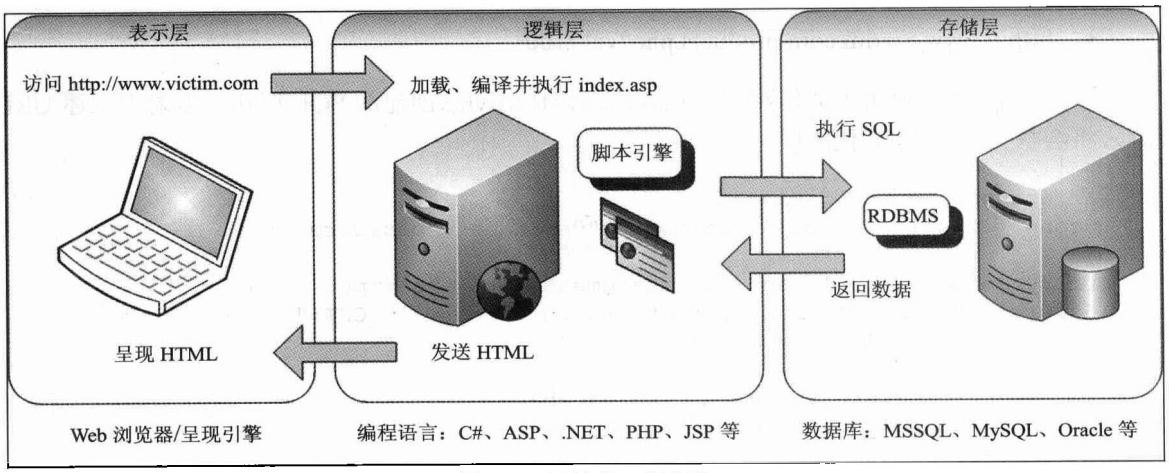


图 1-1 简单的三层架构

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器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器，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 层应用开发范式。其中包括一种四层解决方案，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 描绘了一种简单的四层架构。

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所提供的 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所提供的 API。应用服务器使用数据库连接器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器，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。

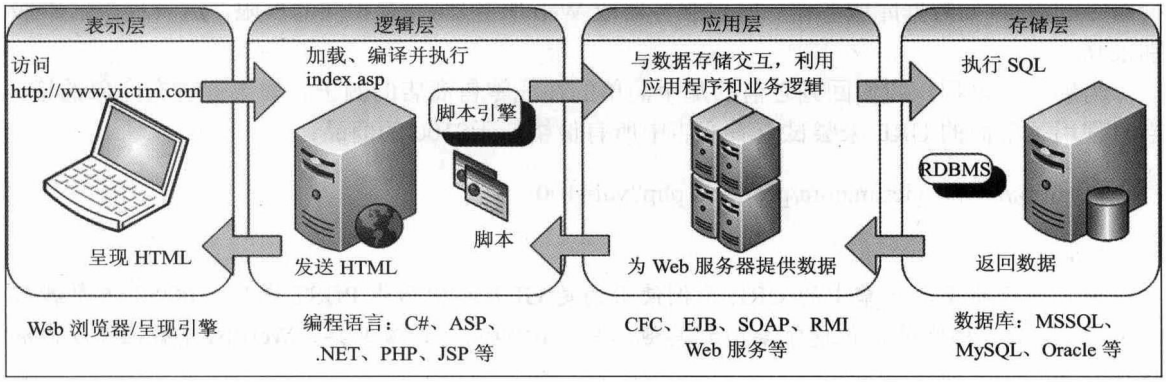


图 1-2　四层架构

层式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。各个层部署在不同的机器上，或者虽然位于同一台机器上，但实际上或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，三层架构和四层架构是最常见的部署架构。正如前面所介绍的，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。

1.3　理解 SQL 注入

Web 应用越来越成熟，技术也越来越复杂。它们涵盖了从动态 Internet 和内部网入口(如电子商务网站和合作企业外部网)到以 HTTP 方式传递数据的企业应用(如文档管理系统和 ERP 应用)。这些系统的有效性及其存储、处理数据的敏感性对于主要业务而言都极其关键(而不仅仅是在线电子商务商店)。Web 应用及其支持的基础结构和环境使用了多种技术，这些技术可能包含很多在他人代码基础上修改得到的代码。正是这种功能丰富的特性以及便于通过 Internet 或内部网对信息进行比较、处理、散播的能力，使它们成为流行的攻击目标。此外，随着网络安全技术的不断成熟，通过基于网络的漏洞来攻破信息系统的机会正不断减少，黑客开始将重心转向尝试危害应用上。

SQL 注入是一种将 SQL 代码插入或添加到应用(用户)的输入参数中的攻击，之后再将这些参数传递给后台的 SQL 服务器加以解析并执行。凡是构造 SQL 语句的步骤均存在被潜在攻击的风险，因为 SQL 的多样性和构造时使用的方法均提供了丰富的编码手段。SQL 注入的主要方式是直接将代码插入到参数中，这些参数会被置入 SQL 命令中加以执行。不太直接的攻击方式是将恶意代码插入到字符串中，之后再将这些字符串保存到数据库的数据表中或将其当作元数据。当将存储的字符串置入动态 SQL 命令中时，恶意代码就将被执行。如果 Web 应用未对动态构造的 SQL 语句所使用的参数进行正确性审查(即便使用了参数化技术)，那么攻击者就很可能会修改后台 SQL 语句的构造。如果攻击者能够修改 SQL 语句，那么该语句将与应用的用户拥有相同的运行权限。当使用 SQL 服务器执行与操行系统交互的命令时，该进程将与执

行命令的组件(如数据库服务器、应用服务器或 Web 服务器)拥有相同的权限，这种权限通常级别很高。

为展示该过程，我们回到之前的那个简单的在线零售商店的例子。如果读者有印象的话，当时使用了下面的 URL 来尝试查看商店中所有价格低于$100 的商品：

- http://www.victim.com/products.php?val=100

注意：

为了便于展示，本章中的 URL 示例使用的是 GET 参数而非 POST 参数。POST 参数操作起来与 GET 一样容易，但通常要用到其他程序，比如流量管理工具、Web 浏览器插件或内联代理程序。

这里我们尝试向输入参数 val 插入自己的 SQL 命令。可通过向 URL 添加字符串*'OR '1'='1*来实现该目的：

- http://www.victim.com/products.php?val=100'OR '1'='1

这次，PHP 脚本构造并执行的 SQL 语句将忽略价格而返回数据库中的所有商品，这是因为我们修改了查询逻辑。添加的语句导致查询中的 OR 操作符永远返回真(即 1 永远等于 1)，从而出现这样的结果。下面是构造并执行的查询语句：

```
SELECT *
FROM ProductsTbl
WHERE Price < '100.00' OR '1'='1'
ORDER BY ProductDescription;
```

注意：

可通过多种方法来利用 SQL 注入漏洞以便实现各种目的。攻击成功与否通常高度依赖于基础数据库和所攻击的互联系统。有时，完全挖掘一个漏洞需要有大量的技巧和坚强的毅力。

前面的例子展示了攻击者操纵动态创建的 SQL 语句的过程，该语句产生于未经验证或编码的输入，并能够执行应用开发人员未预见或未曾打算执行的操作。不过，上述示例并未说明这种漏洞的有效性，我们只是利用它查看了数据库中的所有商品。我们本可以使用应用最初提供的功能来合法地实现该目的。但如果该应用可以使用 CMS(Content Management System，内容管理系统)进行远程管理，会出现什么情形呢？CMS 是一种 Web 应用，用于为 Web 站点创建、编辑、管理及发布内容。它并不要求使用者对 HTML 有深入的了解或者能够进行编码。可使用下面的 URL 访问 CMS 应用：

- http://www.victim.com/cms/login.php?username=foo&password=bar

在访问该 CMS 应用的功能之前，需要提供有效的用户名和口令。访问上述 URL 时会产生如下错误：“Incorrect username or password, please try again”。下面是 login.php 脚本的代码：

```
// connect to the database
$conn = mysql_connect("localhost","username","password");

// dynamically build the sql statement with the input
```

```
$query = "SELECT userid FROM CMSUsers WHERE user = '$_GET["user"]' " .
         "AND password = '$_GET["password"]'";

// execute the query against the database
$result = mysql_query($query);

// check to see how many rows were returned from the database
$rowcount = mysql_num_rows($result);

// if a row is returned then the credentials must be valid, so
// forward the user to the admin pages
if ($rowcount !=0) { header("Location: admin.php");}

// if a row is not returned then the credentials must be invalid
else { die('Incorrect username or password, please try again.')}
```

login.php 脚本动态地创建了一条 SQL 语句。如果输入匹配的用户名和口令，它将返回一个记录集。下列代码更加清楚地说明了 PHP 脚本构造并执行的 SQL 语句。如果输入的 user 和 password 的值与 CMSUsers 表中存储的值相匹配，那么该查询将返回与该用户对应的 userid。

```
SELECT userid
FROM CMSUsers
WHERE user = 'foo' AND password ='bar'
```

这段代码的问题在于应用开发人员相信执行脚本时返回的记录数始终是 0 或 1。在前面的 SQL 注入示例中，我们使用了可利用的漏洞来修改 SQL 查询的含义以使其始终返回真。如果对 CMS 应用使用相同的技术，那么将导致程序逻辑失败。向下面的 URL 添加字符串'OR '1'='1，这次，由 PHP 脚本构造并执行的 SQL 语句将返回 CMSUsers 表中所有用户的 userid。新的 URL 如下所示：

- http://www.victim.com/cms/login.php?username=foo&password=bar 'OR '1'='1

我们通过修改查询逻辑，返回了所有的 userid。添加的语句导致查询中的 OR 操作符永远返回真(即 1 永远等于 1)，从而出现了这样的结果。下面是构造并执行的查询语句：

```
SELECT userid
FROM CMSUsers
WHERE user = 'foo' AND password = 'password' OR '1'='1';
```

应用逻辑是指要想返回数据库记录，就必须输入正确的验证证书，并在返回记录后转而访问受保护的 admin.php 脚本。我们通常是作为 CMSUsers 表中的第一个用户登录的。SQL 注入漏洞可以操纵并破坏应用逻辑。

警告：

不要在任何 Web 应用或系统中使用上述示例，除非已得到应用或系统所有者的许可(最好是书面形式)。在美国，该行为会因违反 1986 年《计算机欺诈与滥用法》(Computer Fraud and Abuse Act of 1986)(www.cio.energy.gov/documents/ComputerFraud-AbuseAct.pdf)或 2001 年《美国爱国者法案》(USA PATRIOT ACT of 2001)而遭到起诉。在英国，则会因违反 1990 年的《计

算机滥用法》(Computer Misuse Act of 1990)(www.opsi.gov.uk/acts/acts1990/Ukpga_19900018_en _1)和修订过的 2006 年的《警察与司法法案》(Police and Justice Act of 2006)(www.opsi.gov.uk/Acts/acts2006/ukpga_20060048_en_1)而遭到起诉。如果控告并起诉成功，那么你将会面临罚款或漫长的监禁。

著名事例

很多国家的法律并没有要求公司在经历严重的安全破坏时对外透露该信息(这一点与美国不同)，所以很难正确且精准地收集到有多少组织曾因 SQL 注入漏洞而遭受攻击或已受到危害。不过，由恶意攻击者发动的安全破坏和成功攻击是当今新闻媒体中一个喜闻乐见的话题。即便是最小的破坏(可能之前一直被公众所忽视)，现在通常也会被大力宣传。

有些公共可用的资源可以帮助理解 SQL 注入问题的严重性。例如，通用漏洞披露组织 CVE(Common Vulnerabilities and Exposures)的 Web 站点上提供了一系列安全漏洞和公布信息，目的在于为众所周知的问题提供统一命名。CVE 的目标是使不同漏洞容器(工具、知识库和服务)间的数据共享变得更容易。该网站整理众所周知的漏洞信息并提供安全趋势的统计分析。在 2007 年的报告中(http://cwe.mitre.org/documents/vuln-trends/index.html)，CVE 共列举了其数据库中的 1754 个 SQL 注入漏洞，其中 944 个是 2006 年新增的。SQL 注入漏洞在 CVE 2006 年报告的所有漏洞中占 13.6% (http://cwe.mitre.org/documents/vuln-trends/index.html)，仅次于跨站脚本(XSS)漏洞，但排在缓冲区溢出漏洞的前面。

此外，开放应用安全计划组织 OWASP(Open Web Application Security Project)在其列举的 2007 年 10 大最流行的影响 Web 应用的安全漏洞中将注入缺陷(包括 SQL 注入)作为第二大漏洞。OWASP 列举出 10 个漏洞的主要目的是让开发人员、设计人员、设计师和组织了解最常见的 Web 应用安全漏洞所产生的影响。OWASP 2007 年公布的 10 大安全漏洞是对 CVE 数据进行精简后汇编而成的。使用 CVE 数据来表示有多少网站受到过 SQL 注入攻击的问题是该数据无法包括自定义站点中的漏洞。CVE 需求代表的是商业和开源应用中已发现的漏洞数量，它们无法反映现实中这些漏洞的存在情况。现实中的情况非常糟糕。

我们还可以参考其他专门整理受损 Web 站点信息的站点所提供的资源。例如，Zone-H 是一个流行的专门记录 Web 站点毁损的 Web 站点。该站点展示了近几年来因为出现可利用的 SQL 注入漏洞而被黑客攻击的大量著名的 Web 站点和 Web 应用。自 2001 年以来，Microsoft 域中的 Web 站点已被破坏过 46 次(甚至更多)。可以在 Zone-H 上在线查看受到攻击的 Microsoft 站点的完整列表(www.zone-h.org/content/view/14980/1/)。

传统媒体同样喜欢大力宣传因数据安全所带来的破坏，尤其是那些影响到著名的重量级公司的攻击。下面是已报道的一些新闻的列表：

- 2002 年 2 月，Jeremiah Jacks 发现 Guess.com(www.securityfocus.com/news/346)存在 SQL 注入漏洞。他因此而至少获取了 200 000 个用户信用卡信息的访问权。
- 2003 年 6 月，Jeremiah Jacks 再次发动攻击，这次攻击了 PetCo.com(www.securityfocus.com/news/6194)，他通过 SQL 注入缺陷获取了 500 000 个用户信用卡信息的访问权。
- 2005 年 6 月 17 日，MasterCard 为保证信用卡系统方案的安全，变更了部分受到破坏的顾客信息。这是当时已知的此种破坏中最严重的一次。黑客利用 SQL 注入缺陷获取了 4 千万张信用卡信息的访问权(www.ftc.gov/os/caselist/0523148/0523148complaint.pdf)。

- 2005 年 12 月，Guidance Software(EnCase 的开发者)发现一名黑客通过 SQL 注入缺陷破坏了其数据库服务器(www.ftc.gov/os/caselist/0623057/0623057complaint.pdf)，导致 3800 位用户的经济记录被泄露。
- 大约 2006 年 12 月，美国折扣零售商 TJX 被黑客攻击，黑客从 TJX 数据库中盗取了上百万条支付卡信息。

以前黑客破坏 Web 站点或 Web 应用是为了与其他黑客组织进行竞赛(以此来传播特定的政治观点和信息)，炫耀他们疯狂的技术或者只是报复受到的侮辱或不公。但现在黑客攻击 Web 应用更大程度上是为了从经济上获利。当今 Internet 上潜伏的大量黑客组织均带有不同的动机。其中包括只是出于对技术的狂热和“黑客”心理而破坏系统的个人，专注于寻找潜在目标以实现经济增值的犯罪组织，受个人或组织信仰驱动的政治活动积极分子以及心怀不满、滥用职权和机会以实现各种不同目的的员工和系统管理员。Web 站点或 Web 应用中的一个 SQL 注入漏洞通常就足以使黑客实现其目标。

您的网站被攻击了么？

这种事不会发生在我身上，是吧？

多年来我评估过很多 Web 应用，在所测试的应用中我发现有三分之一易遭受 SQL 注入攻击。该漏洞所带来的影响因应用而异，但现在很多面向 Internet 的应用中均存在该漏洞。很多应用暴露在不友善的环境中，比如未经过漏洞评估的 Internet。毁坏 Web 站点是一种非常嘈杂、显眼的行为，“脚本小子”[1]通常为赢得其他黑客组织的比分和尊重而从事该活动，而那些非常严肃且目的明确的黑客则不希望自己的行为引起注意。对于老练的攻击者来说，使用 SQL 注入漏洞获取内联系统的访问权并进行破坏是个完美可行的方案。我曾不止一次告诉过客户他们的系统已遭受攻击，目前黑客正利用它们来从事各种非法活动。有些组织和 Web 站点的所有者可能从未了解他们的系统之前是否被利用过或者当前系统中是否已被黑客植入了后门程序。

2008 年年初至今，数十万 Web 站点遭到一种自动 SQL 注入攻击的破坏。该攻击使用一种工具在 Internet 上搜索存在潜在漏洞的应用。如果发现了存在漏洞的站点，该工具使自动利用该漏洞。传递完可利用的净荷(payload)之后，它执行一个交互的 SQL 循环来定位远程数据库中用户创建的每一张表，然后将恶意的客户端脚本添加到表的每个文本列中。由于大多数数据库驱动的 Web 应用使用数据库中的数据来动态构造 Web 内容，因而该脚本最终会展现给受危害的 Web 站点或应用的用户。标签(tag)会指示浏览器加载受感染的 Web 页面，从而执行远程服务器上的恶意脚本。这种行为的目的是让该恶意程序感染更多主机。这是一种非常高效的攻击方式。重要的站点(比如由政府部门维护的站点、联合国和较大公司的站点)均遭受过大量这种攻击的破坏和感染。很难准确地弄清在连接到这些站点的客户端电脑和访问者中有多少受到了感染或破坏(尤其是当发动攻击的个人自己定义传递的可利用净荷时)。

1. 真正的黑客对那些只会模仿且水平低下的年青人的谑称。