# http://127.0.0.1/sqli-labs/Less-1/?id=1

## Host头攻击漏洞

### 漏洞描述

为了方便的获得网站域名，开发人员一般依赖于HTTP Host header，但是这个header是不可信赖的，如果应用程序没有对host header值进行处理，就有可能造成恶意代码的传入。

### 漏洞位置

/sqli-labs/Less-1/?id=1

### 漏洞请求

GET /sqli-labs/Less-1/?id=1 HTTP/1.1  
User-Agent: Mozilla/4.0 (Mozilla/4.0; MSIE 7.0; Windows NT 5.1; FDM; SV1; .NET CLR 3.0.04506.30)  
Accept-Encoding: gzip, deflate  
Accept: \*/\*  
Connection: keep-alive  
Host: www.baidu.com

### 修复建议

web应用程序应该使用SERVER\_NAME而不是host header。

在Apache和Nginx里可以通过设置一个虚拟机来记录所有的非法host header。在Nginx里还可以通过指定一个SERVER\_NAME名单，Apache也可以通过指定一个SERVER\_NAME名单并开启UseCanonicalName选项。

## SQL注入漏洞

### 漏洞描述

Web程序中对于用户提交的参数未做过滤直接拼接到SQL语句中执行，导致参数中的特殊字符破坏了SQL语句原有逻辑，攻击者可以利用该漏洞执行任意SQL语句，如查询数据、下载数据、写入webshell、执行系统命令以及绕过登录限制等。

### 漏洞位置

/sqli-labs/Less-1/?id=1%29%28%22%27

### 漏洞请求

GET /sqli-labs/Less-1/?id=1%29%28%22%27 HTTP/1.1  
Host: 127.0.0.1  
User-Agent: Mozilla/4.0 (Mozilla/4.0; MSIE 7.0; Windows NT 5.1; FDM; SV1; .NET CLR 3.0.04506.30)  
Accept-Encoding: gzip, deflate  
Accept: \*/\*  
Connection: keep-alive

### 修复建议

代码层最佳防御sql漏洞方案：使用预编译sql语句查询和绑定变量：

1、使用预编译语句，使用PDO需要注意不要将变量直接拼接到PDO语句中。所有的查询语句都使用数据库提供的参数化查询接口，参数化的语句使用参数而不是将用户输入变量嵌入到SQL语句中。当前几乎所有的数据库系统都提供了参数化SQL语句执行接口，使用此接口可以非常有效的防止SQL注入攻击。

2、对进入数据库的特殊字符（’”<>&\*;等）进行转义处理，或编码转换。

3、确认每种数据的类型，比如数字型的数据就必须是数字，数据库中的存储字段必须对应为int型。

4、数据长度应该严格规定，能在一定程度上防止比较长的SQL注入语句无法正确执行。

5、网站每个数据层的编码统一，建议全部使用UTF-8编码，上下层编码不一致有可能导致一些过滤模型被绕过。

6、严格限制网站用户的数据库的操作权限，给此用户提供仅仅能够满足其工作的权限，从而最大限度的减少注入攻击对数据库的危害。

7、避免网站显示SQL错误信息，比如类型错误、字段不匹配等，防止攻击者利用这些错误信息进行一些判断。

8、过滤危险字符，例如：采用正则表达式匹配union、sleep、and、select、load\_file等关键字，如果匹配到则终止运行。