你知道 SQL 注入么？

“有人的地方就有江湖，有数据库存在的地方就可能存在 SQL 注入漏洞。”

何谓 SQL 注入？

注： 这里面会涉及到一些数据库操作相关的代码，不懂的同学，可以自行的查找一下数据库的相关操作，也是很简单的。

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 注入其实就是恶意用户通过在表单中填写包含 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「越俎代庖」（yuè zǔ dài páo）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。

SQL 注入实例

接下来我们用代码，来实现 SQL 注入，来看一下，如何实现 SQL 注入攻击，以及 SQL 注入的实质。

接下来展示一个正常的查询操作

studentDao.js

```
const mySql = require('mysql');
// 创建数据库连接对象
const connection = mySql.createConnection({
    host: "127.0.0.1",
    port: "3306",
    user: "root",
    password: "Welcome2duyi",
    database: "school"
});

connection.on('error', err => console.log(err));
```

```
// 创建 sql 语句
let name = "'熊大'"
let sql = "select * from student where name = "

// 数据库连接
connection.connect();

// 进行查询操作
connection.query(sql+name, function (err, result) {
    if(err) {
        throw err
    } else {
        console.log(result)
    }
})

// 数据库断开连接
connection.end();
```

我们通过执行 node studentDao.js 可以得到结果：

```
$ node testDao.js
[ RowDataPacket { id: 2, stu_num: 2, name: '熊大', age: 18, class: 3, pwd: 'asdfgh' } ]
```

数据库内容：

这是正经的数据库操作，但是当我们的 name 是来自用户输入，那他的情况就会很多。

比如下面的操作（向数据库中插入一条数据）

studentDao.js

```
// ...

// 创建 sql 语句（）
```

```
let name = "'Robert');drop table student;"
let sql = "insert into student (name) values("

// 数据库连接
connection.connect();

// 进行查询操作
connection.query(sql+name, function (err, result) {
    if(err) {
        throw err
    } else {
        console.log(result)
    }
})

// 数据库断开连接
connection.end();
```

由于 Node 中的 mysql 做了处理，所以可以避免这样的注入问题，给我们报了一个错误：

```
throw err; // Rethrow non-MySQL errors
^

Error: ER_PARSE_ERROR: You have an error in your SQL syntax; check the manual
that corresponds to your MySQL server version for the
right syntax to use near 'drop table student' at line 1
```

drop student 中存在 关键字，存在危险操作。

如果正常执行，会怎么样？

程序没有暑促任何数据内容，报了一条错误，表单 student 无法找到。

这是为什么呢？问题就在于我们所插入的数据项中包含 SQL 关键字 drop table，这两个关键字的意义是从数据库中清除一个表单。而关键字之前的 Robert');使得 SQL 执行器认为上一命令已经结束，从而使得危险指令 drop table 得到执行。也就是说，这段包含 drop table 关键字的数据项使得原有的简单的插入姓名信息的 SQL 语句

```
"insert into students (name) values ('Robert')"
```

变为了同时包含另外一条清除表单命令的语句

```
"insert into students (name) values ('Robert');DROP TABLE students;--"
```

而 SQL 数据库执行上述操作后，students 表单被清除，因而表单无法找到，所有数据项丢失。

通过下面的动漫图示，来再次说明 SQL 注入。

如何防止 SQL 注入问题

大家也许都想到了，注入问题都是因为执行了数据项中的 SQL 关键字，那么，只要检查数据项中是否存在 SQL 关键字不就可以了么？的确是这样，很多数据库管理系统都是采取了这种看似**方便快捷**的过滤手法，但是这并不是一种根本上的解决办法，如果有个美国人真的就叫做『Drop Table』呢？你总不能逼人家改名字吧。对于这种情况都每种数据库会有不同的解决办法，例如 Node mySql 可以以字符串的形式提交就可以了，like this ：

```
insert into student (name) values('drop table')
```

合理的防护办法有很多。

数据校验

首先，尽量避免使用常见的数据库名和数据库结构。在上面的案例中，如果表单名字并不是 students，则注入代码将会在执行过程中报错，也就不会发生数据丢失的情况——SQL 注入并不像大家想象得那么简单，它需要攻击者本身对于数据库的结构有足够的了解才能成功，因而在构建数据库时尽量使用较为复杂的结构和命名方式将会极大地减少被成功攻击的概率。

其次，使用正则表达式等字符串过滤手段限制数据项的格式、字符数目等也是一种很好的防护措施。理论上，只要避免数据项中存在引号、分号等特殊字符就能很大程度上避免 SQL 注入的发生。

另外，就是使用各类程序文档所推荐的数据库操作方式来执行数据项的查询与写入操作，比如在上述的案例中，如果我们稍加修改，首先使用 execute()方法来保证每次执行仅能执行一条语句，然后将数据项以参数的方式与 SQL 执行语句分离开来，就可以完全避免 SQL 注入的问题。或者进行数据转义也是可以避免 SQL 注入。

权限限制

严格限制 Web 应用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最低权限，从而最大限度的减少注入攻击对数据库的危害。**请记住永远不要使用超级用户或所有者帐号去连接数据库！**当数据库被攻击时将损伤限制在当前表的范围是比较明智的选择。通过权限限制可以防止攻击者获取数据库其它信息，甚至利用数据库执行 Shell 命令等操作。

日志处理

当数据库操作失败的时候，尽量**不要将原始错误日志返回**，比如类型错误、字段不匹配等，把代码里的 SQL 语句暴露出来，以防止攻击者利用这些错误信息进行 SQL 注入。除此之外，在允许的情况下，**使用代码或数据库系统保存查询日志**也是一个好办法。显然，日志并不能防止任何攻击，但定期审计数据库执行日志可以跟踪是否存在应用程序正常逻辑之外的 SQL 语句执行。日志本身没用，要查阅其中包含的信息才行。毕竟，更多的信息总比没有要好。

说在最后

当然，做好数据库的备份，同时对敏感内容进行加密永远是最重要的。某些安全性问题可能永远不会有完美的解决方案，只有我们做好最基本的防护措施，才能在发生问题的时候亡羊补牢，保证最小程度的损失。