MySQL 及 SQL 注入：

所谓 SQL 注入，就是通过把 SQL 命令插入到 Web 表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。

为了防止 SQL 注入，我们需要注意以下几个要点：

（1）永远不要信任用户的输入。对用户的输入进行校验，可以通过正则表达式，或限制长度；对单引号和 双"-"进行转换等；

（2）永远不要使用动态拼装 sql，可以使用参数化的 sql 或者直接使用存储过程进行数据查询存取；

（3）永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接；

（4）不要把机密信息直接存放，加密或者 hash 掉密码和敏感的信息；

（5）应用的异常信息应该给出尽可能少的提示，最好使用自定义的错误信息对原始错误信息进行包装；

（6）sql注入的检测方法一般采取辅助软件或网站平台来检测，软件一般采用 sql 注入检测工具 jsky，网站平台就有亿思网站安全平台检测工具。MDCSOFT SCAN 等。采用 MDCSOFT-IPS 可以有效的防御 SQL 注入，XSS 攻击等。

MySQL 导出数据：

使用 SELECT...INTO OUTFILE 语句导出数据

以下实例中我们将数据表 runoob_tbl 数据导出到 /tmp/runoob.txt 文件中:

```
mysql> SELECT * FROM runoob_tbl
    -> INTO OUTFILE '/tmp/runoob.txt';
```

可以通过命令选项来设置数据输出的指定格式，以下实例为导出 CSV 格式：

```
mysql> SELECT * FROM passwd INTO OUTFILE '/tmp/runoob.txt'
    -> FIELDS TERMINATED BY ',' ENCLOSED BY ""
    -> LINES TERMINATED BY '\r\n';
```

导出 SQL 格式的数据：

如果需要导出整个数据库的数据，可以使用以下命令：

```
$ mysqldump -u root -p RUNOOB > database_dump.txt
password ******
```

如果需要备份所有数据库，可以使用以下命令：

```
$ mysqldump -u root -p --all-databases > database_dump.txt
```

password ******

将数据表及数据库拷贝至其他主机

可以在 mysqldump 命令中指定数据库名及数据表。

在源主机上执行以下命令，将数据备份至 **dump.txt 文件**中：

$ mysqldump -u root -p database_name table_name > dump.txt

password *****

如果**完整备份数据库**，则无需使用特定的表名称。如果需要将备份的数据库导入到 **MySQL 服务器**中，可以使用以下命令，使用以下命令需要确认数据库已经创建：

$ mysql -u root -p database_name < dump.txt

password *****

也可以使用以下命令将导出的数据**直接导入到远程的服务器**上，但请**确保两台服务器是相通的，是可以相互访问**的：

$ mysqldump -u root -p database_name \

| mysql -h other-host.com database_name

MySQL 导入数据：

（1）mysql 命令导入

格式为：

mysql -u 用户名 -p 密码 < 要导入的数据库数据(runoob.sql)

实例：

mysql -uroot -p123456 < runoob.sql

以上命令将备份的整个数据库 runoob.sql 导入。

（2）source 命令导入。source 命令导入数据库需要先登录到数库终端：

```
mysql> create database abc;          # 创建数据库
mysql> use abc;                      # 使用已创建的数据库
mysql> set names utf8;               # 设置编码
mysql> source /home/abc/abc.sql      # 导入备份数据库
```

（3）使用 LOAD DATA 导入数据：

MySQL 中提供了 LOAD DATA INFILE 语句来插入数据。以下实例中将从当前目录中读取文件 dump.txt ，将该文件中的数据插入到当前数据库的 mytbl

表中：

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl;
```

LOAD DATA 默认情况下是按照数据文件中列的顺序插入数据的，如果数据文件中的列与插入表中的列不一致，则需要指定列的顺序：

```
mysql> LOAD DATA LOCAL INFILE 'dump.txt'
    -> INTO TABLE mytbl (b, c, a);
```

（4）使用 mysqlimport 导入数据：

mysqlimport 客户端提供了 LOAD DATA INFILEQL 语句的一个命令行接口。mysqlimport 的大多数选项直接对应 LOAD DATA INFILE 子句。

从文件 dump.txt 中将数据导入到 mytbl 数据表中，可以使用以下命令：

```
$ mysqlimport -u root -p --local mytbl dump.txt
password *****
```

mysqlimport 命令可以指定选项来设置指定格式,命令语句格式如下：

```
$ mysqlimport -u root -p --local --fields-terminated-by=":" \
    --lines-terminated-by="\r\n"    mytbl dump.txt
password *****
```

mysqlimport 语句中使用 --columns 选项来设置列的顺序：

```
$ mysqlimport -u root -p --local --columns=b,c,a \
    mytbl dump.txt
password *****
```

mysqlimport 的常用选项介绍：

选项	功能
-d or --delete	新数据导入数据表中之前删除数据数据表中的所有信息
-f or --force	不管是否遇到错误，mysqlimport将强制继续插入数据
-i or --ignore	mysqlimport跳过或者忽略那些有相同唯一 关键字的行， 导入文件中的数据将被忽略。
-l or -lock-tables	数据被插入之前锁住表，这样就防止了， 你在更新数据库时，用户的查询和更新受到影响。
-r or -replace	这个选项与 - i选项的作用相反；此选项将替代 表中有相同唯一关键字的记录。
--fields-enclosed- by=char	指定文本文件中数据的记录时以什么括起的， 很多情况下 数据以双引号括起。 默认的情况下数据是没有被字符括起的。
--fields-terminated-by=char	指定各个数据的值之间的分隔符，在句号分隔的文件中， 分隔符是句号。您可以用此选项指定数据之间的分隔符。 默认的分隔符是跳格符（Tab）
--lines-terminated-by=str	此选项指定文本文件中行与行之间数据的分隔字符串 或者字符。 默认的情况下mysqlimport以newline为行分隔符。 您可以选择用一个字符串来替代一个单个的字符： 一个新行或者一个回车。