

SQL注入

开发相关的网络攻击及防范

例子：重置密码


`<input type="hidden" name="email" value="user@host"> <input type="hidden" name="code" value="5421">`

`sql = "update app_users set password="" + pwd + "" where email="" + email + "" and code="" + code + """;`

`<input type="hidden" name="email" value="user@host' or '1'='1" >`

`sql: update app_users set password='xxx' where email='user@host' or '1'='1' and code='5421'`

触发条件

- 有通过页面输入或URL传递的参数
- 参数可被客户端修改或依赖客户端输入
- 输入的内容包含了", ' 等SQL中的字符未被转换；也未做其他合法性检查
- 程序拼接了SQL语句，未使用SQL的参数模式 

SQL中的参数 (JDBC)

```
PreparedStatement ps = conn.prepareStatement("update app_user set password=? where email=?  
and code=?");
```

```
ps.setString(1, md5Pwd);  
ps.setString(2, email);  
ps.setString(3, code);
```

```
ps.executeUpdate();
```

SQL中的参数 (Mybatis)

```
<update id="update">  
    update app_user set password=#{md5Pwd} where id=#{id}  
</update>
```

#{ } vs. \${ }

使用参数的另外一个原因：性能

- 数据库会先解析并编译每个SQL，之后执行SQL
- 编译结果会缓存

一个简单表，插入1000条SQL的对比

| | |
|---------|--------|
| 拼接SQL | 3.569秒 |
| 使用SQL参数 | 2.789秒 |

多花费**28%**的时间

程序拼接SQL

- 尽量用SQL传参方式
- 拼接SQL时，要对单引号等特殊字符做转义
- 如果传递进来的sql中包含字段名、表名、统计函数等，一定要限定范围

```
String sql = "delete from " + request.getParamger("table") + " where ..."
```

```
String sql = "select id," + request.getParamger("col") + " from user where ..."
```