# 输入验证

## 前置知识

对于移动应用来说，用户都会涉及到输入数据，然后客户端把用户输入的数据发送到服务端进行处理。如果服务端没有对输入数据进行验证的机制，那么很可能攻击者提交恶意的数据来攻击系统。

**威胁描述：**

客户端把用户输入的数据发送到服务端进行处理。如果数据包含非法字符或者恶意代码，而服务端没有进行恰当的验证就进行处理的话，可能导致系统被攻击（如：SQL注入）。

**涉及功能点：**

登录

## 修复方案

1. 客户端发送到数据发服务端，在服务端处理数据前，先要对数据合规性进行检查。
2. 使用长度方法检查手机号的长度是否满足11位长度；可以使用正则表达式来对手机号进行验证，检查其是否符合手机号的规则要求。
3. 使用长度方法检查密码的长度是否满足8到20位的长度。可以用正则检查密码字符是否至少是字母数字组合。
4. 使用长度方法检查验证码的长度是否满足长度要求；可以使用正则表达式来对验证码进行验证。
5. 也可以使用正则表达式来对verifycode进行验证。

## 代码参考

检查手机号的合规性

|  |
| --- |
| public static boolean isMobile**(**String str**)** **{**  String regExp **=** "^((13[0-9])|(15[^4])|(18[0,2,3,5-9])|(17[0-8])|(147))\\d{8}$"**;**  Pattern p **=** Pattern**.**compile**(**regExp**);**  Matcher m **=** p**.**matcher**(**str**);**  **return** m**.**matches**();**  **}** |

检查密码的合规性

|  |
| --- |
| public static boolean isPassword**(**String str**)** **{**  **if** **(**str**.**length**()** **<=** 8 **&&** str**.**length**()** **>=** 20**)** **{**  **return**  **}**  String regExp **=** "[a-zA-Z0-9]"**;**  Pattern p **=** Pattern**.**compile**(**regExp**);**  Matcher m **=** p**.**matcher**(**str**);**  **return** m**.**matches**();**  **}** |

检查验证码的合规性

|  |
| --- |
| public static boolean isCode**(**String str**)** **{**  String regExp **=** "^[a-zA-Z0-9]{4}$"**;**  Pattern p **=** Pattern**.**compile**(**regExp**);**  Matcher m **=** p**.**matcher**(**str**);**  **return** m**.**matches**();**  **}** |

检查verifycode的合规性

|  |
| --- |
| public static boolean isVerifycode **(**String str**)** **{**  String regExp **=** "^[a-zA-Z0-9]$"**;**  Pattern p **=** Pattern**.**compile**(**regExp**);**  Matcher m **=** p**.**matcher**(**str**);**  **return** m**.**matches**();**  **}** |