IOS客户端通用测试

IOS客户端安全策略设置

4.1 软键盘随机化排布

测试项描述

在使用自定义软键盘进行敏感信息输入的情况下，有可能被恶意软件进行屏幕点击位置劫持，获取对应的点击位置，得到其输入内容。

测试步骤

在输入敏感信息输入框中，如果存在自定义软键盘，在输入框中多次调出键盘，检查键 盘中的字符分布是否是随机的。

结果判定

如图，字符排列随机

风险评级

低风险

安全建议

每次调用键盘时候，需要对键盘进行随机化排布(通常是利用数组表示键排布，则需要对数组进行随机打乱)。

IOS客户端安全策略设置

4.2 密码修改验证

测试项描述

检测是否需要输入旧密码验证来修改密码。

测试步骤

进入密码输入界面，检查密码是否需要输入原密码并进行原密码校验

结果判定

如图所示，需要进行原密码验证来修改密码，视为安全。

风险评级

中风险

安全建议

在程序修改密码界面中添加原密码输入框，并且后台先验证原密码才允许设置新密码

IOS客户端安全策略设置

4.3 密码复杂度检验

测试项描述

通常客户端没有设置禁止修改口令的强弱判断，导致用户有时贪图方便修改成简单密钥使攻击者容易通过弱口令进入帐号进行非法操作。检测客户端修改密码能否输入弱口令。

测试步骤

1.进行修改密码操作，将密码修改成弱口令 123456，会失

2.进行修改密码操作，将密码修改成弱口令，会成功:

结果判定

使用存在连续数字、重复数字密码进行尝试如果可以修改成功即存在风险。

风险评级

低风险

安全建议

程序完全修复工程量非常大，而且很容易有遗漏。建议只在程序中进行重复数字、连续数字、连续字符、关键字符串(admin、test、hahaha)、常见口令(1qaz2wdc、123qwe! @#)之类的密码进行过滤即可。

IOS客户端安全策略设置

4.4 URL 认证失败锁定

测试项描述

测试账户在登陆时候多次输入密码错误或修改密码时候多次错误是否会导致账号被锁定。

测试步骤

多次输入错误密码，检查其是否会将账号锁定。

结果判定

如图所示

会在 5次密码错误后锁定账号，视为安全。

风险评级

低风险

安全建议

后台中，通过维护一张 currentErrorMap(线性安全)，使用用户名作为 key，使用错误次数作为 value;并且重写其 add 方法，检测其错误次数，如果错误次数达到闻值，则写入1到数据库中 isLocked 字段中，禁止登陆。

IOS客户端安全策略设置

4.5 单点登录限制

测试项描述

绝大多数APP都不需要数合设备同时在线，允许同时在线则无法及时得知是否账号已经被攻击者获取登录并进行操作，也无法通过紧急措施将恶意登录的一方挤下线，保护账号 安全。

测试步骤

在设备A中进行登录在设备 B中再登录相同账号

结果判定

在上述设备 A 中进行操作，检查是否能够正常操作。

风险评级

低风险

安全建议

方法一，使用spring security 代码如下:

```
// web.xml
<listener>
    <listener-class>org.springframework.security.web.session.HttpSessionEvent
Publisher</listener-class>
</listener>
<security:http auto-config="true">
    //.....
    <security:session-management>
        <security:concurrency-control max-sessions="1" error-if-maximum-
exceeded="true"/>
    </security:session-management>
    //.....
</security:http>
```

方法二，使用HashMap(原始方法，当然可以用redis替代)

```
User result = userService.login(user.getFuUserName(), user.getFuPassword());
if(result!=null){
    String sessionId = super.getSessionId(false);
    for (String username : loginUserMap.keySet()) {
        if(!username.equals(result.getFuUserName()) || loginUserMap.containsValue(sessionI
d)){
            continue;
        }
        isExist = true;
        break;
    }
    if(isExist){
        super.setRequestAttr(Constant.MESSAGE, "抱歉，该用户已登录！");
        return "error";
    }else {
        loginUserMap.put(result.getFuUserName(), sessionId);
    }
```

效果图

IOS客户端安全策略设置

4.6 界面切换保护

测试项描述

当用户在登录界面写入账号密码之后，若界面切换时候未能清除用户输入的密码，则可 能导致手机丢失后被直接使用已输入的信息进行登录。

测试步骤

进入登录界面，输入账号密码:

测试步骤

进入手机桌面再将程序从后台调出

结果判定

若切换回到程序，密码未消除，则存在风险。参考测试步骤。

风险评级

低风险

界面切换保护

安全建议

```
1   - (void)viewDidLoad {
2   // ......
3       [[NSNotificationCenter defaultCenter]addObserver:self
4                                selector:@selector(clearMyInput)
5                   name:UIApplicationDidEnterBackgroundNotification
6                                object:nil];
7   }
8   - (void)clearMyInput {
9       self.usernameField.text = @"clear";
10      self.passwordField.text = @"clear";
11  }
```

IOS客户端安全策略设置

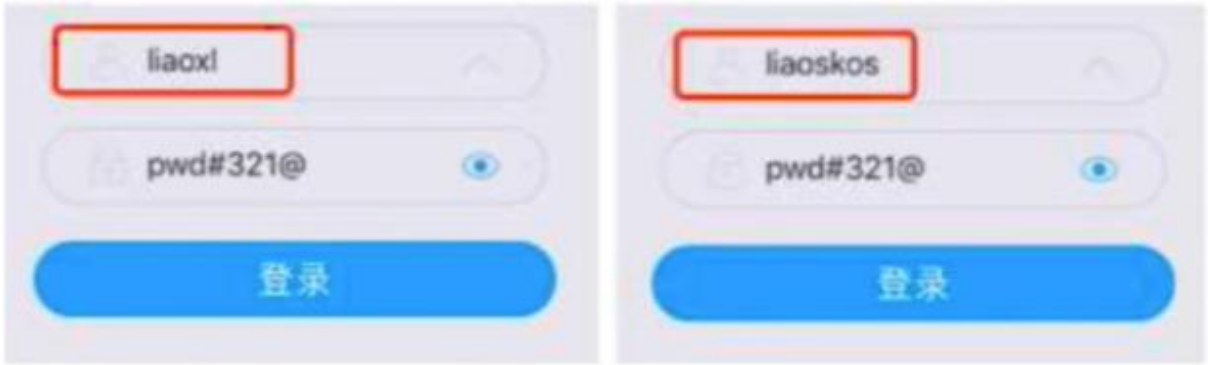
4.7 UI敏感信息安全

测试项描述

登录过程中会为了友好，提示“用户名错误”或若“密码错误”。此时就可以通过这种“友好”的提示检查项目是否存在相关用户名，并且在账号体系不存在锁定的情况下相对容 易地进行密码爆破。

测试步骤

登录界面中输入存在的用户名和请求登录，再使用不存在的用户名进行请求登录:



结果判定

不安全，低风险。如图所示

风险评级

低风险

安全建议

将提示统一修改成“用户名或密码错误”。

IOS客户端安全策略设置

4.8 信息屏蔽检测

测试项描述

检测程序中显示敏感信息是否使用*进行部分屏蔽，防止在被截屏获取敏感信息，同时 也防止他人窥探手机屏幕得到敏感信息。

测试步骤

运行并登陆程序，使用程序中的功能，检查各个功能中个人或者企业隐私信息是否完全 以明文方式显示。

结果判定

存在明文显示的姓名、电话、身份证号、银行卡号等敏感信息都可以判定为不安全若都以*进行部分屏蔽则为安全。

风险评级

低风险

安全建议

在程序返回信息中使用*替代部分敏感信息，客户端只需负责显示。

IOS客户端安全策略设置

4.9 会话超时策略

测试项描述

程序在长时间不进行操作的时候，如果程序没有自动锁定或者退出登录，则有可能导致在这种情况下造成数据窃取。

测试步骤

1.登录应用，并且在20分钟内不进行操作

2.20分钟后再继续操作。

结果判定

若20分钟后能够继续进行内容操作，则存在风险

若20分钟后不再能够继续进行内容操作或者要求输入密码进入操作界面，都是安全的。

风险评级

低风险

安全建议

- 方法一，web.xml

```
<session-config>
    <session-timeout>2</session-timeout>
</session-config>
```

- 方法二，tomcat web.xml

```
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

- 方法三，使用对应的 API

```
HttpSession session = request.getSession();
session.setMaxInactiveInterval(60);//单位为秒
```

1. 方法四，检测点击事件之间的时间长度，或者从点击之后开始计时，一旦这个时间长度大于预设的闲置时间，则由客户端发起退出请求退出登录。

IOS客户端安全策略设置

4.10 安全退出/注销

测试项描述

程序在进行正常退出之后如果后台不消除相关的用户身份凭证，则在身份凭证被窃取之 后容易造成数据泄露。

测试步骤

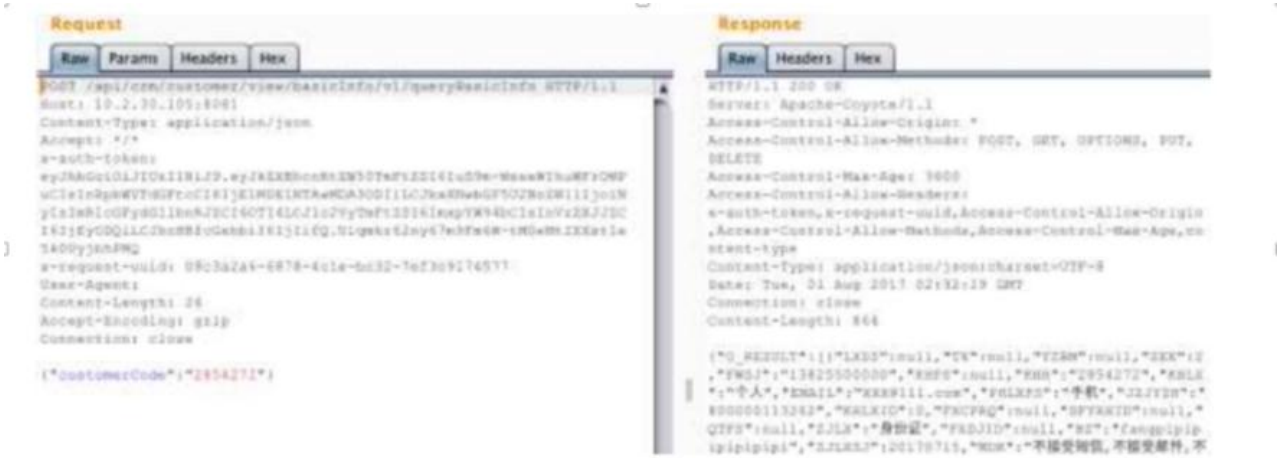
1.数据抓包

2.退出程序(安全退出 和直接返回键退出)

3.将此前的数据包进行重放

结果判定

不安全，中风险。如图所示，使用JWT 进行交互，在退出后长时间内仍然能够获取敏 感数据。



风险评级

中风险

安全建议

谢谢观赏