基于Android应用

登录权限的分析

目录

[1、简介 2](#_Toc483721043)

[2、AOP动态分析 3](#_Toc483721044)

[2.1登录认证 3](#_Toc483721045)

[3、权限漏洞及解决办法 6](#_Toc483721046)

[3.1权限漏洞分析 6](#_Toc483721047)

[3.2解决方案 7](#_Toc483721048)

[4、实验验证 7](#_Toc483721049)

**基于Android应用的登录权限分析**

**摘要：**

随着互联网高速发展，3G/4G时代的到来，智能手机高度普及，使用智能手机的用户数量呈现爆炸式增长。在现有移动终端安装的系统中，Android系统超过七成，Android系统的App种类繁多，应用覆盖范围广泛。其中大多数App都需要与服务器端进行交互，交互的第一步即为登录认证。为了保证用户信息的安全，必须进行权限控制，防止用户信息泄露而造成威胁财产安全。为了寻找App可能存在的登录权限的漏洞，本文总结了八个开源Android项目的登录认证方式。

**关键词**：Android；登录；权限漏洞；AOP

## 1、简介

移动互联网的飞速发展和智能终端的普及，特别是从Android系统问世以来，移动智能设备使用用户呈现爆炸性增长。据《2015 年中国移动互联网研究报 告》显示，截止到 2015 年 12 月，国内在网活跃移动智能 设备数量达到 8.99 亿， 其中，Android 设备占据七成[1]。 与此同时，各大应用商店中的App 数量急剧增长，种类 从办公、娱乐、游戏到购物等应有尽有。

其中大多数App需要与后台服务器进行交互，登录认证是与服务器进行交互的入口，是用户信息安全的第一道防线，也是进行用户权限控制的第一步，因此登录认证的过程显得尤为重要。本文分析了利用AOP动态分析八个开源Android项目获得的数据，总结了现有App在登录认证方面的常见方法，找出其中的漏洞，并提出了相应的解决方案。

分析开源程序采用动态分析方式，利用AOP可以很有效地进行动态分析。

AOP 是一个概念，一个规范，本身并没有设定具体语言的实现，这实际上提供了非常广阔的发展的空间。AspectJ是AOP的一个很悠久的实现，它能够和 Java 配合起来使用。

AspectJ 中有几个必须要了解的概念：

Aspect： Aspect 声明类似于 Java 中的类声明，在 Aspect 中会包含着一些 Pointcut 以及相应的 Advice。

Joint point：表示在程序中明确定义的点，典型的包括方法调用，对类成员的访问以及异常处理程序块的执行等等，它自身还可以嵌套其它 joint point。

Pointcut：表示一组 joint point，这些 joint point 或是通过逻辑关系组合起来，或是通过通配、正则表达式等方式集中起来，它定义了相应的 Advice 将要发生的地方。

Advice：Advice 定义了在 pointcut 里面定义的程序点具体要做的操作，它通过 before、after 和 around 来区别是在每个 joint point 之前、之后还是代替执行的代码。

因为研究过程中所选用的是开源安卓项目，使用Android Studio作为开发环境，所以需要在Android Studio中集成Aspectj，采用注解的方式在Android Studio中使用AOP。设置一个切面，获取到应用程序中与登录相关的类中的切点，编写Aspect类动态获取应用程序的方法调用关系，通过方法调用关系再结合源码，分析应用程序登录的全过程，研究该过程中涉及到的数据，分析该过程是否存在漏洞，结合具体登录过程提出针对该漏洞的解决方案。

## 2、AOP动态分析

### 2.1登录认证

#### 2.1.1客户端-服务器模式

开始

载入验证码，进行登录准备

输入用户名、密码以及验证码

检查输入的合法性

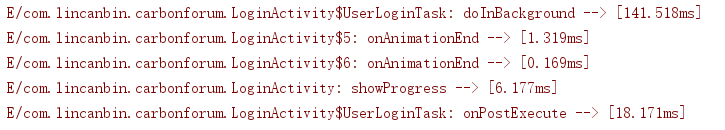
点击登录，向服务器发送数据

服务器返回应答信息

广播程序组件用户信息，更新配置

结束

客户端-服务器模式是最常见的登录模式之一，由客户端向服务器端发送用户登录数据，服务器经过业务处理之后返回给客户端应答信息。如果应答信息表明验证成功，则客户端再次向服务器端请求用户数据，服务器返回用户数据，数据经过客户端解析之后，向应用程序内部各个组件广播用户信息，进行配置更新，完成一次完整的登录过程。



上图为利用AOP获取的程序方法的动态调用过程，可以看出，帐户密码信息发送至服务器端后，服务器端返回相关数据，由onPostExecute方法处理服务器返回的数据。



分析源码可知，用户数据均由服务器端提供，并且认证成功后想程序各个组件广播用户信息，进行配置更新。

#### 2.1.2本地存储

开始

输入用户名和密码

点击登录，向服务器端发送数据

服务器返回验证消息

客户端处理服务器返回的字节流数据

客户端解析成用户数据，获取用户基本资料

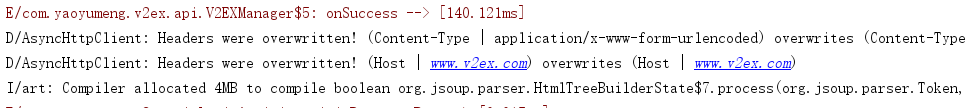
根据用户名读取本地用户数据

应用程序配置用户信息，更新视图

结束

本地读取用户信息的方式，用户输入用户名和密码，然后向服务器端发送数据，等待服务器的认证。如果认证成功，应用程序则根据用户名读取本地数据库中的数据。得到用户数据之后，应用程序广播通知各个组件，更新视图，加载用户私有数据，完成一个完整的登录过程。





在登录过程中调用V2EXManager的相关方法，想服务器发送用户登录数据，从截获的数据可以看到，该应用程序的服务器URL为[www.v2ex.com](http://www.v2ex.com)。

## 3、权限漏洞及解决办法

### 3.1权限漏洞分析

以上两种登录认证方式最为常见，第一种客户端-服务器模式，此类应用通过客户端发送用户名和密码进行认证，认证成功则由服务器数据库返回用户所需的数据，客户端再进行数据配置。这种权限控制方式相对安全，因为无法从客户端突破获取用户数据，只有通过网络或者服务器获取非法数据，从而获得相应权限。但是由于从网络或者服务器端获取数据相对较为困难，要使用一些工具才有可能获取。另外一种常见的漏洞是利用SQL注入，非法访问服务器的数据库数据，不过现在大部分App或服务器都有防范SQL注入的措施，这个漏洞也很难被利用。

第二种本地读取数据的模式，首先由客户端向服务器端发送用户名和密码，等待服务器的认证。如果服务器认证成功，客户端则根据用户名读取本地存放的用户数据，然后进行配置更新。第一阶段为访问数据库以获取身份认证，安全防护较为完备，但是第二阶段读取本地用户数据却没有较好的安全保护。如果在获得服务器认证成功的信息之后，即将读取本地数据时，拦截到用户名信息，并对其进行修改，程序继续运行将会获取到其他用户在本地的数据，进而在客户端配置其他用户的信息。这是一个权限漏洞，用户可以非法访问其他用户的数据。

### 3.2解决方案

针对本地读取用户数据的登录方式，可以通过一些方法提高本地数据的安全性。例如提高访问本地数据的难度，根据用户名建立哈希表，避免用户名直接与本地数据相关联，程序在访问本地数据时，只有正确的用户名才能访问到相应的数据。

## 4、实验验证

（待完成）