XACML的移动应用安全策略及测试方法

曹宛恬 于鹏飞
(全球能源互联网研究院信息通信研究所 南京 210003)
(信息网络安全国网重点实验室 南京 210003)

摘 要 随着移动互联网技术的发展,具有计算功能的移动终端被大量部署,并在大量移动应用的支撑下完成各项任务;愈来愈多的企业允许员工带着他们的个人设备进入工作环境(BYOD模式)。但不同的人员有不同的角色,不同的资源有不同的访问权限,敏感资源一旦被泄露,将可能给企业带来重大的损失。因此,要想全面支持BYOD,保障数据和系统的安全,需要相应移动应用对敏感资源的访问控制进行明确的规定,并在移动应用运行过程中执行。XACML是访问控制策略的统一描述语言,但目前还未见其对移动应用和BYOD的支持。提出基于XACML语言描述移动应用的访问控制策略,研究XACML访问控制策略的测试方法;在此基础上,面向BYOD,针对Android平台上的项目管理APP进行了实例研究,结果展示了所提方法的有效性。

关键词 BYOD,安全,访问控制,XACML,策略
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.11.021

Mobile Application Security Policies and Testing Research on XACML

CAO Wan-tian YU Peng-fei
(Information Communication Research Institute, Global Energy Interconnection Research Institute, Nanjing 210003, China)
(State Grid Key Labratory of Information & Network Security, Nanjing 210003, China)

Abstract With the development of mobile Internet technology, the mobile terminals that have the ability to compute are deployed in great quantities. They can complete various tasks with the support of a large number of mobile applications. More and more companies allow employees to bring their own devices into the work environment, and this can be called BYOD (Bring Your Own Device). But different people have different characters, and different resources have different access permissions. The leak of sensitive resources will lead to significant losses of the enterprise. If BYOD wants to be supported perfectly, it is important to ensure the security of data and system. The access control rules that are defined for access to sensitive resources from the corresponding mobile applications need to be clearly and to be implemented in the running process of mobile applications. XACML is an unified description language of access control policies. Until now, it is unable to support mobile applications and BYOD. In this paper, we proposed a study method of testing XACML policies based on that XACML can describe access control policies of mobile applications. We conducted a case study with a project management app facing BYOD on the Android platform and showed the validity of our method.

Keywords BYOD, Security, Access control, XACML, Policy

1 引言

随着移动互联网技术的发展,移动设备越来越普及,手机(尤其是智能手机)的价格越来越便宜,数据和网络连接越来越便捷,整个世界变得越来越数字化,而移动用户和移动设备也逐渐成为了整个数字世界的主宰。

We Are Social[1] 于2015年1月发布了一份调查报告"Digital, Social & Mobile Worldwide in 2015",该报告针对全球数字、社交和移动等方面做了全面的调查,其中包含了全球240多个国家,还详细分析了30个最大的经济体。

调查报告显示全球移动用户的数量已达到36亿,占全球总人口(72亿)的51%,且年增长率超过5%;全球用户使用移动网络所产生的流量占总网站流量的比例上升到了39%,这也就是说,超过1/3的网站中的网页都可以通过智能手机进行浏览;全球移动连接的数量达到71亿,其中智能手机连接的数量达到27亿,占38%;全球使用手提电脑所产生的网络流量占总网络流量的62%,使用手机所产生的网络流量占总网络流量的31%,使用平板电脑所产生的网络流量占总网

1) 数据来源于 http://wearesocial.net/blog/2015/01/digital-social-mobile-worldwide-2015

到稿日期:2016-03-20 返修日期:2016-06-22 本文受面向电力移动终端的应用测试技术研究(5455HT150029)资助。

曹宛恬(1987-),女,高级工程师,主要研究方向为电力行业信息安全与通信安全,E-mail:caowantian@geiri.sgcc.com.cn;于鹏飞(1989-),男,高级工程师,主要研究方向为电力系统通信技术,E-mail:yupengfei@geiri.sgcc.com.cn。

络流量的 7%[1)]。

从该调查报告可以看出，移动用户和移动设备所占比例很大，且增长趋势明显。其中，智能手机的发展最为突出。

智能手机的操作系统各不相同，表 1 列出了市场调研公司 IDC 在 2015 年初发布的 2014 年全球智能机市场统计报告。

表 1　各个操作系统的智能手机的产出量和市场占有率[1)]

操作系统	2014 年		2013 年		年增长率/%
	产出量/百万部	市场占有率/%	产出量/百万部	市场占有率/%	
Android	10593	81.5	8022	78.7	32.0
iOS	1927	14.8	1534	15.1	25.6
Windows Phone	349	2.7	335	3.3	4.2
BlackBerry	58	0.4	192	1.9	−69.8
其他	77	0.6	23	0.2	234.8
合计	13004	100.0	1018.7	100.0	27.7

根据表 1，智能手机最主流的 4 种操作系统分别是 Android，iOS，Windows Phone 和 BlackBerry，这 4 种操作系统的智能手机的市场占有率之和已经超过了 99%。

在全球范围内，2013 年产出了大约 10 亿部智能手机，而 2014 年大约产出了 13 亿部智能手机，2014 年比 2013 年增长了27.7%。其中，在产出的 13 亿部智能手机中，Android 系统的智能手机大约有 10 亿部，较 2013 年的 8 亿部增长了 32.0%。值得注意的是，Android 系统的市场占有率在 2013 年已高达 78.7%，但 2014 年又以 81.5%的市场占有率创了新高，这一年的年增长率达到了 32%。由此可见，Android 系统早已成为市场占有率最高、最主流的智能手机操作系统。

随着移动互联网技术的发展，移动设备的功能越来越强大，已经足够满足人类日常生活的基本需要，因此，移动设备，尤其是智能手机，凭借其携带方便的特点迅速占领了消费市场，普及率迅速提高，强力地冲击了传统设备市场，人类也越来越依赖移动设备。

近年来，很多企业也逐渐开始允许员工带着自己的移动设备进入工作环境（Bring Your Own Device，BYOD[2]），其一方面可以方便员工的使用、提高员工的工作效率和积极性，另一方面也可以降低企业的设备成本，一举多得。但随之而来的还有诸多的安全隐患，如员工在使用个人设备访问企业的系统、数据等敏感资源的过程中，很有可能会导致敏感资源的泄露。因此，要想安全、可靠地使用 BYOD，必须要有完善的访问控制机制来保护企业的敏感资源。

访问控制机制分为两个部分：访问控制策略、访问控制检查和授权。其中，访问控制策略的主流描述语言是 XACML[3]。XACML 多用于 PC 端，在移动端使用得较少，目前还未见 XACML 对移动应用和 BYOD 的支持。

本文主要从两个角度展开研究：1）探索面向 BYOD、应用 XACML 描述移动应用的访问控制策略，并支持 BYOD 系统中的所有个人设备，包括智能手机、平板电脑、手提电脑等；2）为确保移动应用访问控制策略自身的正确性，研究 XACML 策略的自动测试方法。在此基础上，以智能手机为例，选择目前市场占有率最高的、最主流的 Android 系统，基于一个科研项目管理系统（PORTFOLIO & PROJECT MANAGEMENT），对上述方法进行实例研究。

本文第 2 节介绍技术背景和相关的研究工作，包括访问控制、XACML、Android 平台和应用以及 XACML 策略的测试工具 XPTester[4]；第 3 节介绍本文工作中开发和使用的移动应用实例及其访问控制需求；第 4 节介绍 Android 平台的基于 XACML 策略的科研项目管理系统的设计和实现，包括设计概述、XACML 策略、Android 应用开发，并对访问控制检查和授权进行详细的解释；最后对全文工作进行总结，并介绍未来的工作方向。

2　技术背景和相关研究工作

2.1　访问控制

访问控制[5]是确保数据和系统安全的一种重要手段，防止用户对敏感资源的未授权访问。访问控制的依据便是访问控制策略。

访问控制有一些经典的模型：

（1）基于角色的访问控制（Role Based Access Control，RBAC）[6-8]

针对功能需求，设定了一系列的角色，再把对资源的访问授权给角色，给用户分配不同的角色，当用户访问受保护的资源时，访问控制系统根据用户所属的角色进行授权。

（2）基于属性的访问控制（Attribute Based Access Control，ABAC）[9]

通过属性对访问资源的用户和被访问的资源进行描述，同样，通过环境属性对限制条件进行描述，决策过程中以涉及到的主体（Subject）、资源（Resource）和环境（Environment）的属性为基础来进行授权。

在实现访问控制时，首先需要选择一个访问控制模型，然后根据需求编写访问控制策略，并在代码中实现访问控制授权的逻辑。

2.2　XACML

可扩展的访问控制标记语言[3]（eXtensible Access Control Markup Language，XACML）是由结构化信息标准促进组织（OASIS[10]）批准并发布的。2003 年 2 月，OASIS 发布了 XACML 1.0 版本；到目前为止，已经发展到 XACML 3.0 版本。

2.2.1　XACML 策略的结构

XACML 是一种基于属性的访问控制描述语言，它包含 4 个主要元素：主体（Subject）、动作（Action）、资源（Resource）和环境（Environment）。

XACML 策略的结构如图 1 所示。一个完整的 XACML 策略包含一个策略集，一个策略集中又包含若干个策略集或者策略，一个策略中包含若干条规则。每个策略集、策略和规则都包含唯一的目标，目标描述了主体、动作、资源以及环境之间的约束，限定了包含该目标的策略集、策略或者规则能够适用的访问控制请求。每条规则都定义了一个授权结果

1) 数据来源于 http://tech.ifeng.com/a/20150225/40988386_0.shtml

(Effect),它的值可以是允许(Permit)或者拒绝(Deny),该值表明访问控制请求是否被批准。

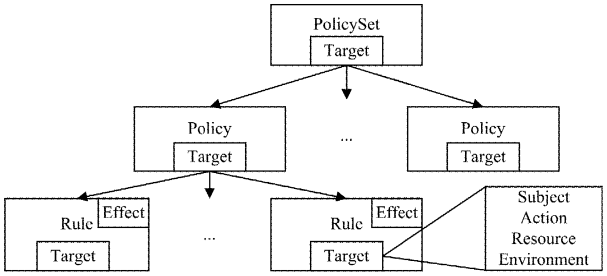

图 1 XACML 策略的结构

2.2.2 XACML 的组合算法

在查询 XACML 策略进行授权评估时,先将 XACML 访问控制请求与 XACML 策略最顶层的策略集中的目标进行匹配,若匹配失败,则返回不适用(Not-Applicable);否则,继续与下面的策略集、策略或者规则中的目标进行匹配,如此递归下去,直至匹配到规则,如果规则匹配成功,则返回规则中的授权结果的值,即允许(Permit)或者拒绝(Deny);否则,返回不适用(Not-Applicable)。

有时针对同一个 XACML 访问控制请求,可能会出现多个规则匹配成功,这会导致不同的授权结果。为了返回唯一的授权结果,XACML 中定义了一些组合算法(Combining Algorithm)作为发生冲突时的评估依据,如 Deny-overrides(拒绝覆盖),Ordered-deny-overrides(有序拒绝覆盖),Permit-overrides(允许覆盖),Ordered-permit-overrides(允许按顺序覆盖),First-applicable(最先应用)[11]。

2.2.3 XACML 的工作流程

基于 XACML 策略的访问控制系统的工作流程如图 2 所示。在应用运行到需要进行访问控制授权的模块时,就需要通过策略实施点(Policy Enforcement Point,PEP)向策略决策点(Policy Decision Point,PDP)发送 XACML 访问控制请求;策略决策点(PDP)接收到 XACML 请求后,查询 XACML 策略并进行授权评估,得到该请求的授权结果,同时将该授权结果返回给策略实施点(PEP),应用根据授权结果作出相应的反应。

图 2 基于 XACML 策略的访问控制系统的工作流程

将 XACML 策略、策略实施点 PEP 和策略决策点 PDP 三者分离,能有效降低系统的耦合度,提高可扩展性和可维护性。

2.3 Android 平台

Android 是由 Andy Rubin 创立的一个基于 Linux 的开源的手机操作系统。Andy Rubin 也被大家称为 Android 之父。

2005 年 8 月,Android 被 Google 公司收购。2007 年 11 月 5 日,Google 发布了 Android 1.0 版手机操作系统[12];到目前为止,Android 的版本已经更新到 6.0[13]。

Android 系统的成长非常迅速,目前已成为市场占有率最高的手机操作系统。

图 3 给出了 Android 系统的体系结构。从图 3 中可以看出,Android 系统主要由 5 个部分组成。

(1)应用程序层

Android 系统包含一系列自带的核心应用程序,包括桌面、联系人、通话、浏览器等。

(2)应用程序框架

Android 应用程序框架为 Android 应用的开发者提供了大量的 API。

(3)函数库

Android 包括 9 个子系统,分别是:图层管理、媒体库、SQLite、OpenGLEState、FreeType、WebKit、SGL、SSL 和 libc。

Android 应用的开发者不能直接调用这些库,但可以通过它上面的应用程序框架提供的接口来进行调用。

(4)Android 运行时

Android 运行时由两部分组成:Android 核心库集和 Dalvik 虚拟机。其中,Android 核心库集能提供 Java 语言核心库的绝大部分功能,Dalvik 虚拟机用来运行 Android 应用程序[12]。

(5)Linux 内核

Android 系统建立在 Linux 2.6 之上[12]。

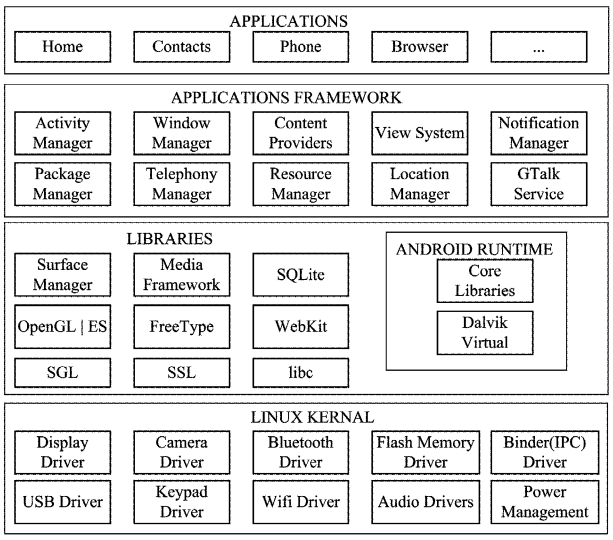

图 3 Android 系统的体系结构[1)]

2.4 Android 应用

2.4.1 Android 应用的源代码

Android 应用的源代码大致分为如下 3 类。

(1)界面布局文件

以 XML 文件的形式定义,文件中每个标签都对应于相应的 View 标签。

(2)Java 源文件

Android 应用的 4 大组件(活动(Activity)、服务(Servi-ce)、内容提供商(Content Provider)和广播接收器(Broadcast Receiver))都是采用 Java 代码实现的。

1) 图来源于 http://mobile.51cto.com/aprogram-446245_all.htm

(3)资源文件

主要以各种 XML 文件为主，其中还可以包括一些图片资源，如 *.png，*.jpg，*.gif 等形式的文件[12]。

2.4.2　Android 应用的四大组件

Android 应用中有四大组件：活动(Activity)、服务(Service)、内容提供商(Content Provider)和广播接收器(Broadcast Receiver)。这四大组件是 Android 应用中至关重要的组成部分。每一种组件都有其特定的功能和生命周期，并定义这个组件怎样创建和销毁。

(1)活动(Activity)

简单地说，活动就是 Android 应用的界面。一个 Activity 通常代表了一个单独的屏幕，开发者可以在 Activity 上部署一些控件(如按钮、输入框等)，并对控件进行处理，也可以监听某些事件(如按钮的点击等)并作出响应。

(2)服务(Service)

没有图形界面，在后台运行，主要用来执行长期运行的操作或远程连接的操作。

(3)内容提供商(Content Provider)

用来管理应用的数据的共享集。应用的数据可以存储在文件系统、SQLite 数据库、网络上等应用能够访问的任何持久且稳定的存储位置。Content Provider 主要用于应用数据的对外共享，并把应用数据共享给其他应用，即如果 Content Provider 允许，其他应用也可以通过 Content Provider 查询甚至修改应用数据。

(4)广播接收器(Broadcast Receiver)

用来响应整个系统范围内的广播公告。Broadcast Receiver 虽然不显示用户界面，但可能会通过创建状态栏通知来提醒用户广播事件的发生。

2.4.3　Activity

活动(Activity)就是 Android 应用的界面，用来与用户进行交互。一个应用可以只含有一个 Activity，但通常应用都会含有多个 Activity，其中有一个 Activity 会被指定为主 Activity。每一个 Activity 都可以启动另一个 Activity 来执行与自己不同的操作。每次一旦有新的 Activity 启动，之前的 Activity 就会被停止，并被系统保存到回退栈中。当新的 Activity 启动时，它也同时被压入回退栈中，并获取用户的焦点。回退栈遵循基本的后进先出的栈机制，因此当用户完成了与当前的 Activity 的交互并按返回键时，当前的 Activity 会从回退栈中弹出并被销毁，同时之前的 Activity 得以恢复。

(1)Activity 的状态

Activity 有 3 种基本状态，以 Activity_A 为例进行说明。

1)运行(Resumed 或者 Running)

Activity_A 在前台并且拥有用户的焦点。

2)暂停(Paused)

记另一个 Activity 为 Activity_B，其在前台并且拥有焦点，但 Activity_A 仍然是可见的。也就是说，Activity_B 是可见的并且在 Activity_A 之上，Activity_B 是部分透明的或者没有完全覆盖整个屏幕。

Paused 的 Activity 完全是活跃(alive)的，即 Activity 对象保存在内存中，维护着所有的状态和成员信息，仍然连接着窗口管理器，但是在内存极低的情况下有可能会被杀死。

3)停止(Stopped)

Activity_A 完全被另一个 Activity 掩盖。

Stopped 的 Activity 也仍然是活跃(alive)的，即 Activity 对象被保存在内存中，维护着所有的状态和成员信息，但是不连接窗口管理器，而且不再对用户可见，当内存不够用时，有可能会被杀死。

如果一个 Activity 是 Paused 的或者 Stopped 的，系统则有可能会把它从内存中移除：要么调用该 Activity 的 finish()方法，要么直接杀死进程。

(2)Activity 的生命周期

伴随着 Activity 在不同状态之间的转换，会有一些基本的生命周期回调方法，如onCreate()，onStart()，onResume()，onPause()，onStop()，onDestroy()。所有的这些回调方法都能被重写，可以用来在 Activity 状态改变时执行合适的操作。

如图 4 所示，这些生命周期回调方法定义了一个 Activity 的整个生命周期，通过实现这些方法可以监控 Activity 生命周期的 3 个嵌套循环。

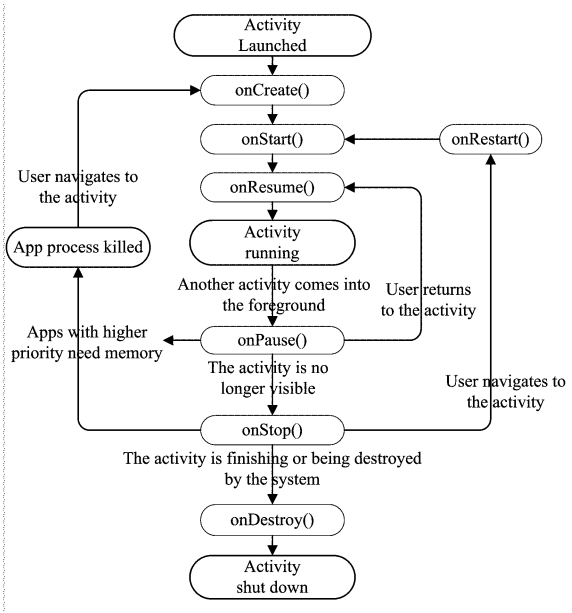

图 4　Activity 的生命周期[1)]

(1)整个生命周期

简单来说，整个生命周期是指 Activity 从创建到被杀死的整个过程，即从调用 onCreate()方法开始，到调用 onDestroy()方法结束。Activity 应该在开始时，即在 onCreate()方法中，设置全局状态(如定义界面布局)；在结束时，即在 onDestroy()方法中，释放所有的资源。

(2)可视生命周期

这个 Activity 发生在调用 onStart()方法和调用 onStop()方法之间，对用户可见，用户可以在屏幕中看到这个 Activity 并且与之交互。在 Activity 的整个生命周期内，随着 Ac-

1) 图来源于 http://developer.android.com/guide/components/activities.html

tivity 状态的变化，系统可能会多次调用 onStart()方法和 onStop()方法。

(3)前台生命周期

这个 Activity 发生在调用 onResume()方法和调用 onPause()方法之间，不仅对用户可见，而且拥有用户的焦点，在屏幕上所有的其他 Activity 之前。为了避免状态切换时速度缓慢而出现用户等待的情况，onResume()和 onPause()这两个方法中的代码应该是轻量级的。

2.4.4 Service

服务(Service)是 Android 应用的四大组件之一，不提供用户界面，主要用来在后台执行长时间运行的操作或者执行远程连接的工作。

(1)Service 的形式和生命周期

1)被启动的(Started)

一个组件(如 Activity)通过调用 startService()方法启动 Service，此时，一个 Started 的 Service 被创建。通常情况下，Started 的 Service 执行一个单一操作，并不会向它的启动者返回任何结果。

启动 Service 的组件销毁，并不影响 Started 的 Service 在后台的运行，除非这个 Service 自身调用的 stopSelf()方法停止或者别的组件调用 stopService()方法将其停止，否则它会无限期地一直运行在后台。如果 Started 的 Service 停止了，系统就会将其销毁。

2)被绑定的(Bound)

一个组件(如 Activity)通过调用 bindService()方法绑定到 Service，此时，一个 Bound 的 Service 被创建。Bound 的 Service 提供了一种客户端-服务器端接口 IBinder。将 Bound 的 Service 作为服务器，允许绑定组件。绑定到这个 Service 的组件可以通过 IBinder 接口与 Service 交互，向 Service 发送请求，获得 Service 返回的响应，甚至可以跨进程进行通信(IPC)。

只有存在别的组件绑定到 Serivce 时，Bound 的 Service 才能运行。客户端可以调用 unbindService()方法关闭连接。Bound 的 Service 作为服务器，可以被多个客户端绑定，因此可以有多个组件绑定到同一个 Service。这些绑定到同一个 Service 的所有组件都解除绑定时，这个 Service 就会被系统销毁。

事实上，Service 可以同时以以上 2 种形式工作，它既可以被启动(无限期地运行)，也可以允许绑定，只要实现 2 个回调方法：onStartCommand()方法和 onBind()方法，其中 onStartCommand()方法允许启动，而 onBind()方法允许绑定。

除了上面所说的情况，Service 也有可能出现其他情况而被杀死。当内存很低，必须占用部分内存为当前拥有用户焦点的 Activity 恢复系统资源时，Android 系统会强制停止并杀死 Service。

Service 也有一些生命周期回调方法，如 onCreate()，onStartCommand()，onBind()，onUnbind()，onRebind()，onDestroy()。这些方法都可以被重写，用来监控 Service 状态的变化，并在适当的时候执行一些操作。

图 5 给出了 Service 的生命周期。图 5(a)显示了被调用 startService()方法创建的 Service 的生命周期，图 5(b)显示了被调用 bindService()方法创建的 Service 的生命周期。

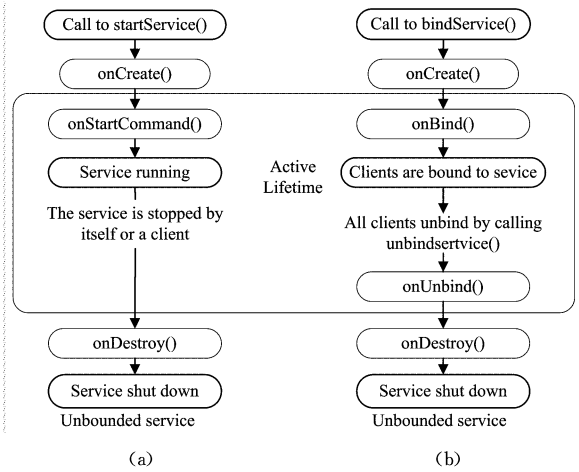

图 5 Service 的生命周期[1)]

通过实现图 5 中的生命周期回调方法，可以监控 Service 的生命周期中的 2 个嵌套循环。

1)整个生命周期

整个生命周期指 Service 从被创建到被销毁的整个过程，即从调用 onCreate()方法开始，到调用 onDestroy()方法结束。Service 在 onCreate()方法中进行初始设置，在onDestroy()方法中释放资源。所有的 Service 都需要调用 onCreate()方法和 onDestroy()方法，不管这个 Service 是被调用 startService()方法创建的还是被调用 bindService()方法创建的。

2)活跃生命周期

活跃生命周期是从调用 onStartCommand()方法或者调用 onBind()方法开始的。这两种方法分别将 Intent 传递给 startService()方法和 bindService()方法。如果一个 Service 是 Started 的，那么一旦它的活跃生命周期结束，它的整个生命周期都将随之结束。如果一个 Service 是 Bound 的，那么当它的 onUnbind()方法返回时，它的活跃生命周期便结束。

虽然图 5 分别展示了 Service 被调用 startService()方法创建和被调用 bindService()方法创建的情况，但是对任何 Service 而言，不管它何时被启动，都允许绑定客户端。

(2)Service 的绑定

一个 Bound 的 Service 提供了一种客户端-服务器端接口 IBinder，Service 作为服务器，允许绑定组件。绑定到 Service 的组件可以通过 IBinder 接口与 Service 交互，向Service发送请求，获得 Service 返回的响应，甚至可以跨进程进行通信(IPC)。有 3 种定义接口的方式：

1)扩展 Binder 类

如果 Service 是对应用来说是私有的，并且和客户端运行在相同的进程内，那么可以通过扩展 Binder 类并在 onBind()方法中返回 Binder 实例的方式创建通信接口。客户端接收

1) 图来源于 http://developer.android.com/guide/components/services.html

这个 Binder，可以用它直接访问在 Binder 甚至 Service 中定义的公有(public)方法。

2)使用信使(Messenger)

如果需要跨越不同的进程使用通信接口，即执行进程间的通信(IPC)，可以使用 Messenger 为 Service 创建接口，但这种通信方式只能传送消息(Message)。在这种方式中，Service 定义了一个 Handler 来对不同类型的 Message 对象作出响应，这个 Handler 允许客户端使用 Message 对象并发送给服务器。另外，客户端也可以定义一个 Messenger，这样服务器 Service 就可以同样发回消息。

3)使用 AIDL

AIDL(Android Interface Definition Language)执行进程间的通信(IPC)时，会将对象分解为操作系统能够理解的基本单元并按次序排列。事实上，上面提到的使用信使(Messenger)的技术也是基于 AIDL 实现的，它使用了 AIDL 作为它的底层结构，Messenger 在单线程中创建了一个包含所有客户端请求的队列，因此 Service 每次只能接收一个请求。但是使用 AIDL 技术能够使 Service 同时处理多个客户端请求。

2.4.5　Android 应用的数据存储方式

Android 提供了几种保存持久、稳定的应用数据的方式。

(1)SharedPreferences

用键值对的方式存储私有的简单数据，保存的数据主要是类似于配置信息格式的数据。

(2)文件存储

文件存储包括内部存储(Internal Storage)和外部存储(External Storage)。内部存储就是在设备内存中存储私有数据；外部存储就是在共享的外部存储中存储公有数据。

(3)SQLiteDatabase

使用 Android 提供的 SQLiteDatabase 数据库来存储结构化的数据，产生的数据库是私有的。

(4)网络连接

把数据存储在自己的网络服务器上。

2.4.6　Android 应用的安全架构

Android 有自己的安全架构，其设计的核心思想是：在默认情况下，任何应用都不能执行任何对其他应用、操作系统或者用户造成不利和损失的操作，例如：读写用户的敏感数据(如联系人、电子邮件)、读写别的应用的文件、执行网络连接等。

因为每个应用都运行在进程沙箱中，应用必须明确地分享资源和数据，所以需要为沙箱所不能提供的额外功能声明权限。应用静态声明运行需要的所有权限，在应用被安装时，Android 系统会提示用户并希望获得用户的同意。Android 没有动态授权的机制。

对于一个 Android 应用而言，存在两种情况需要声明权限：1)这个 Android 应用需要被授权才能调用 Android 系统的功能；2)这个 Android 应用可以被其他应用调用，因此需要声明其他应用调用它所需要的权限[12]。

不管怎样，一个应用若想访问受保护的功能，则必须在 AndroidManifest.xml 清单文件中对相应的权限进行声明。当这个应用被安装到设备上时，安装程序通过检查应用的签名证书和询问用户来决定是否授权所请求的权限，如果获得了用户的许可，应用可以使用受保护的功能。如果应用运行需要的权限并没有被声明，那么应用在运行过程中尝试访问受保护的功能时，会直接失败，不会为用户发送任何通知。应用也可以使用权限保护自己的组件，包括 Activity，Service，Broadcast Receiver 和 Content Provider。

2.5　XPTester

XPTester[4]是基于符号执行的 XACML 策略的测试工具，运行在 Linux 下。XPTester 的界面如图 6 所示。

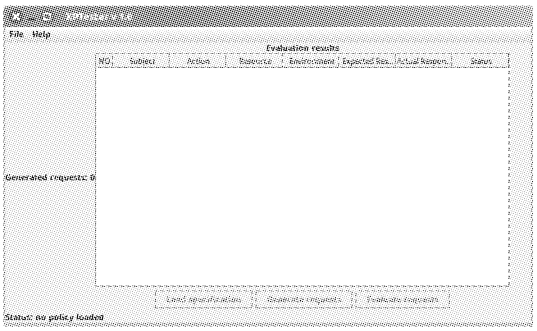

图 6　XPTester 界面

使用 XPTester 时，只需载入被测试的 XACML 策略文件，XPTester 就可以自动地生成 XACML 测试请求，并且对测试结果进行评估。另外，如果用户提供了对访问控制需求的描述，XPTester 还可以对 XACML 测试请求匹配的结果进行自动判断，并记录失败的测试。

XPTester 的核心思想，即对 XACML 策略进行测试的流程，分为 4 个步骤：

Step1　将被测试的 XACML 策略转换成语义上等价的 C 语言代码。

Step2　运用符号执行技术，针对 Step1 中得到的 C 语言代码生成 C 语言的测试输入。

Step3　将 Step2 中得到的 C 语言的测试输入翻译成 XACML 测试请求。

Step4　将 Step3 中得到的 XACML 测试请求代入 XACML 策略中进行匹配和评估，从而获得响应结果，并判断结果是否符合预期。

3　移动应用实例及其访问控制需求

本节详细介绍本文工作中开发和使用的移动应用实例及其访问控制需求。

3.1　移动应用实例

在本文工作中，开发和使用的是一个科研项目管理系统(PORTFOLIO & PROJECT MANAGEMENT，PPM)。使用 PPM 能够更有效地管理项目，帮助规划及监测整个企业中的项目。

PPM 可以允许用户注册、登录并管理与自己相关的项目申请，包括添加或删除项目合作人、修改或删除项目申请、审核项目申请、提交或撤回项目申请等。当然，还允许某些身份的用户申请科研项目。科研项目可以是个人项目，也可以是

几个人合作的项目，科研项目的申请可以经过多次修改，最后提交给相关人员进行审核，若审核全部通过，科研项目则申请成功。

其中主要涉及到的对象是用户和项目申请。

用户分为 4 种角色：Researcher，Chair，Staff 和 Director。

项目申请（proposal）的结构分为两个部分：封面（coversheet）和内容（content）。其中，封面（coversheet）包括：项目负责人（PI）、项目合作人（co-PI）、标题（title）、程序（program）、截止日期（deadline）、总预算（total budget）和持续时间（duration）。

围绕一个项目申请的主要工作流为：

（1）Researcher 创建一个申请并提交，同时成为该项目的负责人。

（2）项目负责人可以添加合作人，合作人最多只能有 4 个。

（3）项目负责人可以修改项目申请。

（4）项目合作人可以修改项目申请的内容部分。

（5）项目负责人可以将项目申请提交给 Chair 进行审核。

（6）在 Chair 审核之后将项目申请交给 Director 审核。

（7）如果审核通过，Staff 将项目发布；否则，将项目申请从系统中撤销。

3.2 访问控制需求

这个科研项目管理系统涉及到一些访问控制需求，即不同的角色或身份具有不同的权限，如表 2 所列。

表 2 不同角色或身份所对应的权限

角色或身份	权限
Researcher	可以创建项目申请
项目负责人	可以添加 Researcher（包括 Chair）作为项目合作人 不能添加 Staff 或者 Director 作为项目合作人 可以删除项目合作人 可以编辑封面和内容 可以删除项目申请 可以将项目申请提交给 Chair 审核 可以提交给 Staff 可以将项目申请提交给 Director 审核，但是要在该项目申请被 Chair 批准之后
项目合作人	不能编辑封面 可以编辑内容 不能添加或者删除项目合作人 不能删除项目申请
Chair	可以批准项目申请，但是要在得到项目负责人的授权之后 不能编辑或者删除项目申请
Director	可以批准或者不批准项目申请 可以删除没通过审核的项目申请 可以撤回已经发布的项目申请
Staff	可以发布通过审核的项目申请

4 移动端访问控制策略的设计及实现与测试

本节介绍 Android 平台的基于 XACML 策略的科研项目管理系统的设计和实现，包括设计概述、XACML 策略、Android 应用开发，并对访问控制的检查和授权进行了详细的解释。

4.1 设计概述

实现针对 Android 系统的 XACML 策略，也就是要实现一个完整的基于 XACML 策略的访问控制的 Android 应用系统，本文实现的是一个科研项目管理系统 PPM。

一个完整的基于 XACML 策略的访问控制系统包括 3 个组成部分：XACML 策略本身、策略实施点 PEP 和策略决策点 PDP，需要分别对这 3 个部分进行实现。

XACML 策略，也就是要按照 XACML 策略的结构和标准，并根据需求进行编写，生成 XML 形式的 XACML 策略文件。

简单地说，策略实施点 PEP 就是在整个 Android 应用系统中生成 XACML 请求并发送给策略决策点 PDP，获得 PDP 响应的授权结果并解析执行的地方，因此需要实现一个访问控制的 Android 应用。

如何实现策略决策点 PDP 呢？首先拟将 XACML 策略转化为 Android 系统能够识别的形式，PDP 模块就由 Android 系统本身提供，但 Android 应用的权限定义在 AndroidManifest.xml 清单文件中，形式很简单，一个功能对应一个权限，无法进行稍复杂的访问控制，因此这种方式不可行。因此，不对 XACML 策略进行转化，而直接对它进行读取，这样就需要自己实现这个 PDP 模块。Android 应用中的 PEP 需要发送 XACML 请求给 PDP，PDP 查询 XACML 策略对 XACML 请求进行评估从而得到授权结果，并将授权结果返回给 PEP，因此 PEP 和 PDP 两者之间需要进行通信，那么可以用 Bound 的 Service 实现 PDP 模块。为了降低耦合，可以将这个 Service 以远程 Service 的形式定义在另一个 Android 应用中。另外，PDP 模块的核心功能，也就是查询 XACML 策略对 XACML 请求进行评估得到授权结果的功能，可以使用开源的 PDP（如 Balana[14]）来实现。实现 PDP 模块后，需要根据 PDP 模块中提供的接口对 PEP 作相应的修改和实现。除此之外，还需要对 XACML 策略进行测试。

因此，整个实现的流程如下：

Step1 编写 XACML 策略。

Step2 开发 Android 应用 PPM，实现科研项目管理系统的功能。

Step3 开发 Android 应用 MyPDP，在 MyPDP 中使用远程 Service 实现 PDP 模块，即策略决策功能。

Step4 根据 MyPDP 中提供的接口修改和实现 PPM 中的 PEP。

Step5 对 XACML 策略进行测试。

整个 Android 应用系统的概览如图 7 所示。

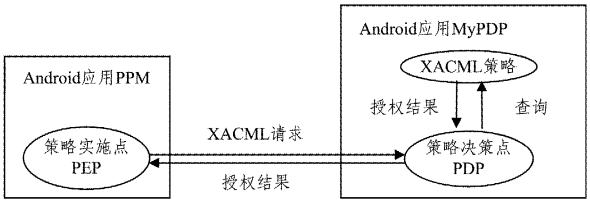

图 7 整个 Android 应用系统的概览

4.2 具体实现

4.2.1 编写 XACML 策略

根据访问控制需求可以得到 27 条规则，这 27 条规则的 4 个主要元素（主体（Subject）、动作（Action）、资源（Resource）、环境（Environment））和授权结果 Effect 如表 3 所列，

其中环境 Environment 相当于条件，表明 proposal 所处的状态。proposal 共有 5 种状态：Created，ApprovedByChair，ApprovedByDirector，DisapprovedByDirector 和 OfficiallySubmitted。

表 3　根据访问控制需求分析得到的 27 条规则

Subject	Action	Resource	Environment	Effect
Researcher	Create	Proposal		Permit
PI	Add	Co-PI	Created	Permit
PI	AddCoPI	Researcher	Created	Permit
PI	AddCoPI	Chair	Created	Permit
PI	AddCoPI	Staff		Deny
PI	AddCoPI	Director		Deny
PI	Delete	Co-PI	Created	Permit
PI	Edit	Coversheet	Created	Permit
PI	Edit	Content	Created	Permit
PI	Delete	Proposal	Created	Permit
PI	SubmitTo	Chair	Created	Permit
PI	SubmitTo	Staff		Permit
PI	SubmitTo	Director	ApprovedByChair	Permit
Co-PI	Edit	Coversheet		Deny
Co-PI	Edit	Content	Created	Permit
Co-PI	Add	Co-PI		Deny
Co-PI	Delete	Co-PI		Deny
Co-PI	Delete	Proposal		Deny
Chair	Approve	Proposal	Created	Permit
Chair	Edit	Coversheet		Deny
Chair	Edit	Content		Deny
Chair	Delete	Proposal		Deny
Director	Approve	Proposal	ApprovedByChair	Permit
Director	Disapprove	Proposal	ApprovedByChair	Permit
Director	Delete	Proposal	DisapprovedByDirector	Permit
Director	Withdraw	Proposal	OfficiallySubmitted	Permit
Staff	Submit	Proposal	ApprovedByDirector	Permit

将这 27 条规则编写为 XACML 策略，并保存成 PPM.xml 文件。由于 XACML 策略具有一定的结构和标准，编写时可以使用开源的 XACML 策略编辑器，如 UMU-XACML-Editor[15]，以避免语法错误。UMU-XACML-Editor 的界面如图 8 所示。

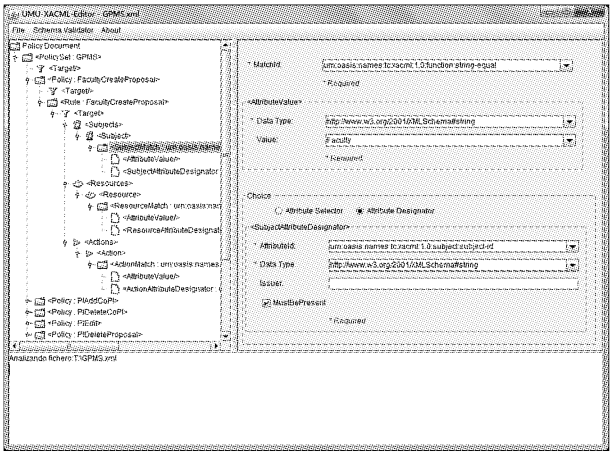

图 8　UMU-XACML-Editor 界面

一个完整的 XACML 策略包含一个策略集，一个策略集中又包含若干个策略集或者策略，一个策略包含若干条规则。每个策略集、策略和规则都包含唯一的目标，目标描述了主体、动作、资源以及环境之间的约束，限定了包含该目标的策略集、策略或者规则能够适用的访问控制请求。每条规则都定义了一个授权结果，它的值可以是允许(Permit)或者拒绝(Deny)。在 Subject 是 Researcher，Action 是 Create，Resource 是 Proposal 的情况下，以 Effect 是 Permit 为例，生成的一条规则如图 9 所示。

```
〈Rule Effect="Permit" RuleId="ResearcherCreateProposal"〉
  〈Target〉
    〈Subjects〉
      〈Subject〉
        〈SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"〉
          〈AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"〉Researcher〈/AttributeV-alue〉
          〈SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="http://www.w3.org/2001/XMLSchema#string" Must-BePresent="true"/〉
        〈/SubjectMatch〉
      〈/Subject〉
    〈/Subjects〉
    〈Resources〉
      〈Resource〉
        〈ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"〉
          〈AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"〉Proposal〈/AttributeValue〉
          〈ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#string" Must-BePresent="true" /〉
        〈/ResourceMatch〉
      〈/Resource〉
    〈/Resources〉
    〈Actions〉
      〈Action〉
        〈ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"〉
          〈AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string"〉Create〈/AttributeValue〉
          〈ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action=id" DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true" /〉
        〈/ActionMatch〉
      〈/Action〉
    〈/Actions〉
  〈/Target〉
〈/Rule〉
```

图 9　规则示例

图 9 给出的规则 Rule 定义了授权结果 Effect 的值是 Permit，并包含了一个目标 Target，目标 Target 包含 3 个元素：主体 Subject 的值是 Researcher、资源 Resource 的值是 Proposal、动作 Action 的值是 Create。当然，如果需要，目标 Target 也可以包含环境 Environment。

4.2.2　开发 Android 应用 PPM

Android 应用 PPM 需要实现的是科研项目管理系统的功能，即允许用户注册、登录并管理与自己相关的项目申请，包括添加或删除项目合作人、修改或删除项目申请、审核项目申请、提交或撤回项目申请等。当然，还允许 Researcher 申请科研项目。科研项目可以是个人项目，也可以是几个人合作的项目，科研项目的申请可以经过多次修改，最后提交给 Chair 和 Director 进行审核，审核全部通过后科研项目申请成功。

根据需求分析，Android 应用 PPM 主要涉及到的对象是用户和项目申请，因此需要为这两者建立数据库。使用 SQLiteDatabase 实现了数据库 ppm_db，其中包含两个表 user_table 和 proposal_table，它们分别存储用户和项目申请的信息。

如表 4 所列，表 user_table 存储的是所有用户的 ID(account_id)、密码(password)、角色(role)和相关的项目申请(proposal_list)。其中，用户的 ID、密码和角色都是简单的字符串形式，而与用户相关的项目申请(proposal_list)则是用这些项目申请的_id(项目申请表 proposal_table 中的自动增长的项)表示的。由于与同一个用户相关的项目申请可能有多个，因此可以将这些项目申请的_id 先加到一个 JSONArray 对象中，再将这个 JSONArray 对象转换成字符串的形式，然后再存储到数据库中。从数据库中读取与某个用户相关的项目申请时，将读取到的 proposal_list 字符串形式转换成 JSONArray 对象，使用 JSONArray 对象的 get()方法依次得到其中的项目申请的_id。

表 4 user_table

列名	数据类型	备注
_id	integer	自动增长
account_id	text	用户的 ID
password	text	用户的密码
role	text	用户的角色
proposal_list	text	与用户相关的项目申请

如表 5 所列，表 proposal_table 存储的是所有项目申请的 ID(proposal_id)、负责人(pi)、合作人(co_pi)、标题(title)、程序(program)、截止日期(deadline)、总预算(total budget)、持续时间(duration)、内容(content)、创建日期(create_date)、状态(status)和相关的用户(user_list)。其中，项目申请的 ID(proposal_id)用创建时间来表示，形如 201501012018；项目的负责人(pi)使用负责人的 ID(account_id)表示；项目的合作人(co_pi)也使用合作人的 ID(account_id)表示，但合作人可能有多个，也使用 JSONArray 来处理；与项目申请相关的用户(user_list)使用用户的 ID(account_id)表示，与同一个项目申请相关的用户也有多个，同样使用 JSONArray 来处理。此外，其余的都是简单的字符串形式。

表 5 proposal_table

列名	数据类型	备注
_id	integer	自动增长
proposal_id	text	用项目申请的创建时间来表示，形如：201501012018
pi	text	项目的负责人，用 account_id 表示
co_pi	text	项目的合作人
title	text	项目的标题
program	text	项目的程序
deadline	text	项目的截止日期，形如：2015-6-6
total_budget	text	项目的总预算，以元为单位
duration	text	项目的持续时间，以天为单位
content	text	项目的内容
create_date	text	项目申请的创建日期，形如：2015-1-1
status	text	项目申请的状态，有 5 种：Created；ApprovedByChair；ApprovedByDirector；DisapprovedByDirector；OfficiallySubmitted
user_list	text	与项目申请相关的用户

在 Android 应用 PPM 中，共设计并实现了 7 个界面：登录界面(MainActivity)、注册界面(SignUpActivity)、项目申请列表界面(ProposalListActivity)、创建项目申请界面(CreateProposalActivity)、项目申请细节界面(ProposalDetailActivity)、编辑封面界面(EditCoversheetActivity)、编辑内容界面(EditContentActivity)。界面的跳转关系如图 10 所示。

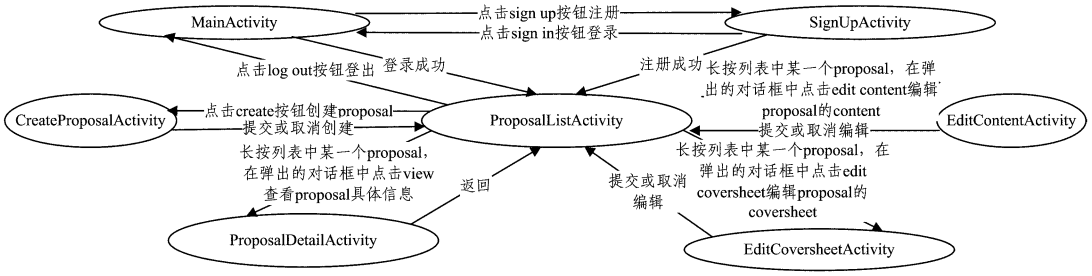

图 10 界面的跳转关系

打开 PPM，进入主界面，即 MainActivity，输入自己的 ID 和密码后点击 sign in 按钮进行登录，登录成功后跳转到 ProposalListActivity；如果还没有账户，也可以直接点击 sign up 按钮跳转到 SignUpActivity 进行注册，注册时除了输入 ID、密码和确认密码外，还需要在 Researcher，Chair，Staff 和 Director 中选择一个角色，然后点击 sign up 按钮进行注册，注册成功后跳转到 ProposalListActivity；当然，如果不小心进入了 SignUpActivity 后发现自己已经有账户，则可以点击 sign in 按钮跳转回 MainActivity 进行登录。

ProposalListActivity 上显示了自己的 ID 和角色，以及与自己相关的项目申请的列表，另外，还有 create 按钮和 log out 按钮。点击 create 按钮，进入 CreateProposalActivity，按照要求输入 title，program，deadline，total budget，duration 和 content 后点击 submit 按钮提交，点击 cancel 按钮取消创建项目申请，不管提交、取消还是返回，都会回到 ProposalListActivity。点击 log out 按钮进行登出，会重新进入 MainActivity。

ProposalListActivity 上显示的与自己相关的项目申请列表只显示了项目申请的简略信息，包含 title，pi 和 status，长按任意一个项目申请，会弹出一个对话框，其中有 11 个选项：view，add co-pi，delete co-pi，edit coversheet，edit content，delete，approve，disapprove，submit to，officially submit，withdraw。

点击 view 进入 ProposalDetailActivity，在这里可以看到这个项目申请的具体信息，返回即可回到 ProposalListActivity；

点击 add co-pi，在弹出的文本对话框中输入想要增加的 co-pi 的 ID，并点击 ok 按钮；

点击 delete co-pi，在弹出的多选对话框中勾选想要删除的 co-pi 的 ID，并点击 ok 按钮；

点击 edit coversheet 进入 EditCoversheetActivity，可以对原来的封面信息（包括 title，program，deadline，total budget，duration）进行修改，点击 submit 按钮提交，点击 cancel 按钮取消修改封面，不管提交、取消还是返回，都会回到 ProposalListActivity；

点击 edit content 进入 EditContentActivity，在这里可以对原来的内容进行修改，点击 submit 按钮提交，点击 cancel 按钮取消修改内容，不管提交、取消还是返回，都会回到 ProposalListActivity；

点击 delete 删除该项目申请；

点击 approve 表明该项目申请通过审核；

点击 disapprove 表明该项目申请没有通过审核；

点击 submit to，在弹出的文本对话框中输入想要提交的项目申请给 Chair 或 Director 或 Staff 的 ID，并点击 ok 按钮；

点击 officially submit 发布该项目申请；

点击 withdraw 撤回该项目申请。

4.2.3　开发 Android 应用 MyPDP

Android 应用 MyPDP 主要需要实现策略决策点 PDP 的功能，即接收 Android 应用 PPM 中的策略实施点 PEP 发送的 XACML 请求，将这个 XACML 请求代入到 XACML 策略中进行评估得到授权结果，并将授权结果返回给 PEP。

也就是说，PPM 需要和 MyPDP 进行通信。由于它们是两个不同的 Android 应用，在两个不同的进程中，因此可以在 MyPDP 中定义一个远程 Service 来实现，这里定义的远程 Service 为 PDPRemoteService。PDPRemoteService 需要允许绑定，因为只有 Bound 的 Service 才能提供通信功能。

PPM 中的 PEP 需要向 PDPRemoteService 发送 XACML 请求，PDPRemoteService 接收到 XACML 请求后，查询 XACML 策略对 XACML 请求进行评估，得到授权结果，并将授权结果返回给 PEP，PEP 接收到授权结果后作出相应的处理。这里需要通信的数据，即 XACML 请求和授权结果，都是字符串类型，因此可以使用消息（Message）来传递数据，即可以使用信使（Messenger）技术实现通信。

而 PDPRemoteService 的核心功能，即策略决策功能，可以使用 Balana[14] 来实现。

由于 Balana 需要知道 XACML 策略所在目录的路径，因此将编写好的 XACML 策略放到该应用中。而作为资源文件，XACML 策略存储在 assets 目录下，只能通过 AssetManager 以二进制流的形式读取，因此要想得到 XACML 策略所在目录的路径，只能将 XACML 策略读取出来并复制到一个固定的路径下。由于 XACML 策略不应该被别的应用修改或者人为修改，因此可以将 XACML 策略复制到 MyPDP 的私有文件目录（/data/data/包名/files）下，而且，由于 XACML 策略只需要复制一次，因此需要在代码中做相应的标记。

图 11 给出了在 PDPRemoteService 中对消息（Message）的处理。PDPRemoteService 中定义了一个信使（Messenger），Messenger 定义了一个 Handler 对 Message 对象进行处理，接收并绑定到它的客户端发送的消息，从中得到 XACML 请求，调用 Balana 中的方法查询 XACML 策略进行评估得到授权结果，将该授权结果也通过 Message 对象发送给客户端指定的接收响应的信使（Messenger）。

```
private Messenger messenger=new Messenger(new handler());
class handler extends Handler {
  @Override
  public void handleMessage(Message msg) {
    //super. handleMessage(msg)
    xacml_request=msg. getData(). getString("request",null);
    if(xacml_request!=null) {
      xacml_decision=getXACMLDecision(xacml_request);
      message=new Message();
      bundle=new Bundle();
      bundle. putString("decision",xacml_decision);
      message. setData(bundle);
      try {
          msg. replyTo. send(message);
      } catch (RemoteException e) {
          e. printStackTrace();
      }
    }
  }
}
```

图 11　在 PDPRemoteService 中对消息（Message）的处理

4.2.4　实现 PPM 中的 PEP

实现 MyPDP 之后，根据 MyPDP 中的 PDPRemoteService 提供的接口来修改和实现 PPM 中的 PEP。PEP 需要绑定到 MyPDP 中的 PDPRemoteService，将 XACML 请求发送给 PDPRemoteService，然后得到 PDPRemoteService 的响应结果，并根据响应结果做出正确的反应。

以点击 PPM 中的项目申请列表界面 ProposalListActivity 上的 create 按钮创建项目申请为例，PEP 对访问控制进行检查和授权的过程分为如下 3 个部分。

（1）绑定到 PDPRemoteService

在 create 按钮的点击事件的监听器中，需要创建一个 createServiceConnection 连接绑定到 MyPDP 中的 PDPRemoteService，如图 12 所示。

```
createServiceConnection=new CreateServiceConnection();
intent=new Intent();
intent. setAction("MyPDP. PDPRemoteService");
bindService(intent, createServiceConnection, Context. BIND_AUTO_CREATE);
```

图 12　绑定到 PDPRemoteService

（2）生成 XACML 请求，并发送给 PDPRemoteService

在与 PDPRemoteService 的连接 createServiceConnection 中，需要根据 PDPRemoteService 提供的 IBinder 接口 service 创建信使 remoteService，然后根据当前的情况生成 XACML 请求，并将该字符串形式的 XACML 请求放到消息 message 中，然后定义处理 PDPRemoteService 对消息 message 的响应的信使，最后使用信使 remoteService 将消息 message 发送给 PDPRemoteService，如图 13 所示。

```
class CreateServiceConnection implements ServiceConnection {
  @Override
  public void onServiceConnected(ComponentName name, IBinder service) {
    remoteService=new Messenger(service);
    xacmlRequest=new XACMLRequest();
    request = xacmlRequest. createXACMLRequest (role,"Proposal",
    "Create");
    message=new Message();
    bundle=new Bundle();
    bundle. putString("request",request);
    message. setData(bundle);
    message. replyTo=new Messenger(new createHandler());
    try {
        remoteService. send(message);
    } catch (RemoteException e) {
        e. printStackTrace();
    }
  }
  @Override
  public void onServiceDisconnected(ComponentName name) {
  }
}
```

图 13 与 PDPRemoteService 的连接

(3)接收授权结果,做出正确的反应

图 14 给出了处理 PDPRemoteService 响应的 createHandler,即对 XACML 请求的授权结果进行解析并作出反应,对 PDPRemoteService 发送过来的消息中得到的授权结果进行判断,若授权结果不是 Permit,则跳出用户没有创建项目申请权限的提示;否则,允许用户创建项目申请,并跳转到创建项目申请界面 CreateProposalActivity。

```
class createHandler extends Handler {
  @Override
  public void handleMessage(Message msg) {
    //super. handleMessage(msg);
    decision=msg. getData(). getString("decision",null);
    System. out. println(decision);
    if(decision!=null) {
      if(!decision. equals("Permit")) {
        new AlertDialog. Builder(ProposalListActivity. this)
            . setTitle("Sorry")
            . setMessage("Sorry, you are not authorized to create proposals!")
            . setPositiveButton("ok",null)
            . show();
        }else {
          intent = new Intent(ProposalListActivity. this, CreateProposalActivity. class);
          startActivity(intent);
          }
        }
    }
}
```

图 14 处理 PDPRemoteService 响应的 CreateHandler

4.3 XACML 策略测试

本文使用工具 XPTester[4] 对 XACML 策略进行测试。使用 XPTester 可以生成 XACML 测试请求,并代入到 XACML 策略中进行评估。另外,如果载入了访问控制需求描述文件,则可以判断评估结果是否符合预期。

4.3.1 编写访问控制需求描述文件

根据访问控制需求分析得到的 27 条规则,编写共计 27 条的访问控制需求描述文件 requirement,表明对各种 XACML 访问控制请求的预期,如图 15 所示。每条描述文件的形式为:主体(Subject) 动作(Action)资源(Resource) 环境(Environment) 预期(Expectation)。如果主体(Subject)、动作(Action)、资源(Resource)、环境(Environment)4 个元素中某个元素没有定义,则用 NewValue 表示,代表任意值。预期(Expectation)用 0 或 1 表示,0 代表 Permit,1 代表 Deny。

```
Researcher Create Proposal NewValue 0
PI Add Co-PI Created 0
PI AddCoPI Researcher Created 0
PI AddCoPI Chair Created 0
PI AddCoPI Staff NewValue 1
PI AddCoPI Director NewValue 1
PI Delete Co-PI Created 0
PI Edit Coversheet Created 0
PI Edit Content Created 0
PI Delete Proposal Created 0
PI SubmitTo Chair Created 0
PI SubmitTo Staff NewValue 0
PI SubmitTo Director ApprovedByChair 0
Co-PI Edit Coversheet NewValue 1
Co-PI Edit Content Created 0
Co-PI Add Co-PI NewValue 1
Co-PI Delete Co-PI NewValue 1
Co-PI Delete Proposal NewValue 1
Chair Approve Proposal Created 0
Chair Edit Coversheet NewValue 1
Chair Edit Content NewValue 1
Chair Delete Proposal NewValue 1
Director Approve Proposal ApprovedByChair 0
Director Disapprove Proposal ApprovedByChair 0
Director Delete Proposal DisapprovedByDirector 0
Director Withdraw Proposal OfficiallySubmitted 0
Staff Submit Proposal ApprovedByDirector 0
```

图 15 访问控制需求描述文件 requirement

4.3.2 运行 XPTester 得到测试结果

运行 XPTester,先载入 XACML 策略文件 PPM. xml,再载入访问控制需求描述文件 requirement,然后生成测试请求,共计 67 个。生成的这 67 个测试请求中每一个测试请求对应 2 个文件,一个是无后缀名的文件,保存的是这个 XACML 请求的 4 个元素:主体、环境、动作、资源;另一个是 XML 文件,保存的是一条规范的 XACML 请求。

生成的测试请求共计 67 个,而访问控制需求描述文件 requirement 中只定义了 27 种情况;对于 40 种没有在访问控制需求描述文件 requirement 中定义的情况,需要手动点击输入框,并在跳出的下拉框中选择预期结果。

最后,对这 67 个测试请求进行评估,结果全部通过,如图 16 所示。图 16 中显示的前 27 个测试请求对应的是访问控制需求描述文件 requirement 中定义的 27 种情况,说明 PPM. xml 访问控制策略是正确的。

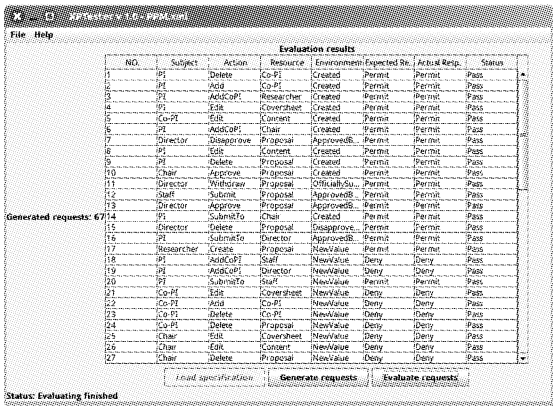

图 16　测试结果

结束语　随着移动互联网技术的发展，移动设备的功能越来越强大，已经足够满足人类日常生活的基本需要，因此，移动设备，尤其是智能手机，凭借其携带方便的特点迅速占领消费市场，普及率迅速提高，强力地冲击了传统的设备市场，人类也越来越依赖于移动设备。

近年来，很多企业逐渐开始允许员工带着自己的移动设备进入工作环境（Bring Your Own Device，BYOD）。员工在使用个人设备访问企业的系统、数据等敏感资源的过程中，很有可能会导致敏感资源的泄露。不同的人员有不同的角色，不同的资源有不同的访问权限，一旦敏感资源泄露，可能会给企业带来重大的损失。因此，要想全面支持 BYOD，必须要有完善的访问控制机制来保障数据和系统的安全。其中，移动应用的访问控制至关重要。

访问控制机制分为两个部分：访问控制策略、访问控制检查和授权。其中，主流的访问控制策略描述语言是 XACML。XACML 是一种基于属性的访问控制描述语言，目前还没有支持移动应用的 XACML，XACML 策略本身的正确性也难以保证。本文提出了基于 XACML 描述移动应用访问控制策略并对 XACML 策略进行自动化测试的方法。

本文在 Android 平台上实现了一个完整的基于 XACML 策略的科研项目管理系统。根据需求分析，对 XACML 访问控制系统的 3 个部分（XACML 策略、策略实施点 PEP、策略决策点 PDP）分别进行了实现。

首先，使用开源的 XACML 策略编辑器 UMU-XACML-Editor[15] 编写 XACML 策略，生成了 XACML 策略文件 PPM. xml。

接着，开发 Android 应用 PPM，实现了科研项目管理系统的功能，允许用户注册、登录并管理与自己相关的项目申请，包括添加或删除项目合作人、修改或删除项目申请、审核项目申请、提交或撤回项目申请等。当然，还允许 Researcher 申请科研项目。科研项目可以是个人项目，也可以是几个人合作的项目，科研项目的申请可以经过多次修改，最后提交给 Chair 和 Director 进行审核，审核全部通过后科研项目申请成功。

然后，开发 Android 应用 MyPDP，在 MyPDP 中定义了一个远程 Service：PDPRemoteService。PDPRemoteService 需要允许绑定，接收 XACML 请求，查询 XACML 策略进行评估得到授权结果，并能将授权结果发回，而查询 XACML 策略进行评估得到授权结果的功能使用了开源的 PDP 和 Balana[14] 进行实现。

最后，根据 MyPDP 中的 PDPRemoteService 提供的接口修改和实现了 PPM 中的 PEP，PEP 绑定到 MyPDP 中的 PDPRemoteService，生成 XACML 请求发送给 PDPRemoteService，并根据 PDPRemoteService 的响应结果做出反应。

另外，还使用测试工具 XPTester[4] 对 XACML 策略进行了测试。

在此后的工作中，可以对本文的工作做进一步的研究和拓展。

（1）本文只对 XACML 策略进行了测试，还没有对应用系统进行测试，可以考虑生成可运行的脚本，对应用系统进行自动化的测试。

（2）目前这个基于 XACML 策略的科研项目管理系统是不联网的，可以把该系统做成联网的形式，并且对它的访问控制做更加完善的规定，比如：可以对网络环境进行限制，在访问某些资源或执行某些操作时，必须使用内网；不仅对访问资源的用户进行限制，而且对访问资源的设备也进行限制。

（3）该系统目前是针对 Android 平台的，可以尝试将该系统部署到更多的平台（如 iOS）上。

（4）更进一步地，结合上面两点，将该基于 XACML 策略的科研项目管理系统扩展成基于网络的客户端-服务器端的形式，实现在所有安装该客户端软件的设备（可以是多种不同的平台）访问资源时进行访问控制，从而实现 XACML 策略对 BYOD 的支持。

参 考 文 献

[1] We Are Social[OL]. http://wearesocial.net.

[2] BYOD: Bring your own device[OL]. http://www.ibm.com/mobilefirst/us/en/bring-your-own-device/byod.html.

[3] RISSANEN E. extensible access control markup language (xacml) version 3.0[OL]. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html.

[4] XPTester: XACML Policy Tester[OL]. http://seg.nju.edu.cn/XPTester.

[5] SANDHU R S, SAMARATI P. Access control: principle and practice[J]. Communications Magazine, IEEE, 1994, 32(9): 40-48.

[6] FERRAIOLO D F, SANDHU R, GAVRILA S, et al. Proposed NIST standard for role-based access control[J]. ACM Transactions on Information and System Security (TISSEC), 2001, 4(3): 224-274.

[7] FERRAIOLO D, KUHN D R, CHANDRAMOULI R. Role-based access control[M]. Artech House, 2003.

[8] SANDHU R S, COYNE E J, FEINSTEIN H L, et al. Role-based access control models[J]. Computer, 1996, 29(2): 38-47.

[9] GOYAL V, PANDEY O, SAHAI A, et al. Attribute-based encryption for fine-grained access control of encrypted data[C]// Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006: 89-98.

[10] OASIS[OL]. https://www.oasis-open.org/cn.

[11] XML 安全：使用 XACML 控制信息访问[OL]. http://www.ibm.com/developerworks/cn/xml/x-xacml.

[12] 李刚. 疯狂 Android 讲义（第 2 版）[M]. 北京：电子工业出版社，2013.

[13] Android Developers[OL]. http://developer.android.com/index.html.

[14] "Balana" The Open source XACML 3.0 implementation[OL]. http://xacmlinfo.org/2012/08/16/balana-the-open-source-xacml-3-0-implementation.

[15] UMU-XACML-Editor[OL]. http://sourceforge.net/projects/umu-xacmleditor.