【科技前沿】TECHNICAL FRONTIER

Android中权限提升漏洞的动态防御技术

张 一，施 勇，薛 质

（上海交通大学信息安全工程学院，上海 200240）

摘 要：在Android操作系统中，权限机制是一项重要的安全机制，它为Android上的应用规定了访问权限，受到了越来越多的关注。文中首先分析了权限的工作机制和它存在的缺陷，然后介绍了权限提升攻击发起的原理，在此基础上提出了动态跟踪防御方法，可以对应用进程间通信（IPC）进行监测，监测是否发生恶意的权限提升。最后，针对Android典型应用进行了实验仿真，结果表明了检测防御方法的有效性。

关键词：Android；权限提升；进程间通信；动态防御

中图分类号：TN929　　**文献标志码**：A　　**文章编号**：1009-8054（2013）11-0071-04

Dynamic Protection Technology for Privilege Escalation Attack on Android

ZHANG Yi, SHI Yong, XUE Zhi

(School of Information Security, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: Permission-based security policy plays a very important role in Android system, and specifies various access privileges for the application of Android. This paper analyzes the principle and defect of permission-based security policy, and then gives the cause of privilege escalation on Android. And based on this, it proposes a new dynamic protection model, thus to implement the supervision of IPC (Inter-Process Communication) policy and prevent the privilege escalation on Android. Finally, the experiment on the Android platform indicates that the dynamic protection is feasible and effective.

Key words: Android; privilege escalation; IPC communication; dynamic protection

0 引言

目前，Android是使用最为广泛的智能手机操作系统之一，随着智能手机的不断推广，Android系统的安全性也越来越受到人们的关注。而权限机制则是整个Android安全机制的核心部分之一，通过为每个应用设置相应的API访问权限，来确保系统的安全性，这是权限机制的核心思想。但是这种机制的权限设置粒度太粗，只有是与否两种情况，导致应用与应用之间的访问权限很开放[1]。所以，利用这一漏洞，一些没有API访问权限的应用可以通过其他获得权限的应用去访问API，这就是所谓的权限提升攻击。而对于这种攻击，可以通过对IPC（进程间通信）的监测，将调用的API和运行的应用权限进行对比，从而找出利用权限提升的应用和进程。

收稿日期：2013-08-12

作者简介：**张一**，1988年生，男，硕士研究生，研究方向为手机安全；**施勇**，1979年生，男，讲师，研究方向为网络与信息安全；**薛质**，1971年生，男，教授，研究方向为信息安全。

1 权限机制

权限机制是一种控制应用访问权限的机制。当一个应用要访问某个 API 时，它就需要有相应的权限才可以访问。比如某个应用需要用到 Android 手机中的 GPS 资源，那它就必须要有 ACCESS COARSE LOCATION 权限或者 ACCESS FINE LOCATION 权限才可以调用相应的 GPS 资源[2]，否则 Android 操作系统就会报安全性异常或者直接不提供相应的 GPS 资源。

除了系统所定义的权限外，Android 中的应用也可以自定义权限来决定其他应用是否有访问的权限。每个应用程序可以对 Activity、Service、Broadcast Receiver、Content Provider 这 4 个组件定义相应的权限访问控制权[3]，这 4 个组件之间的访问关系如图 1 所示。但事实上，由于开发者往往并不考虑应用的安全性，再加上 Android 平台上的应用数量很大，很难有针对性地定义相应的权限，所以事实上，绝大多数的 Android 应用是可以相互访问的。

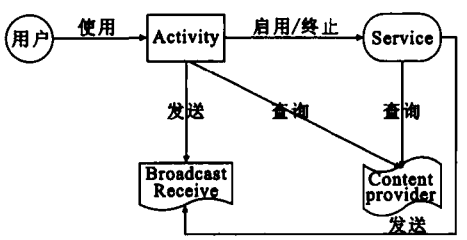

图 1 Android 各组件之间的访问关系

2 权限提升攻击的原理

权限机制只是考虑到单个应用的访问权限[4]，同时又开放应用间的权限定义，这就使得几个应用的组合可以让权限机制形同虚设，从而使得应用的权限得以拓展，导致权限提升攻击。权限提升攻击的核心思想如图 2 所示，应用 A 的组件 a 拥有访问 API C 的权限，而应用 B 则不具备访问 API C 的权限，但是应用 B 却拥有访问应用 A 的组件 a 的权限，这样一来，应用 B 就可以通过应用 A 来实现对 API C 的访问和调用。这就是 Android 的权限提升漏洞，利用这一漏洞，攻击者可以轻易地植入恶意代码，对 Android 用户进行攻击。

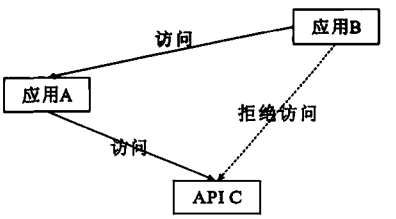

图 2 权限提升攻击原理

目前，Android 权限提升攻击广泛存在，是进行远程攻击的主要方法之一，其主要原因就是权限的定义由开发者来制定，而开发者由于对安全的考虑往往不够充分，导致一些权限的定义存在安全上的隐患，让其他进程和应用可以轻易地访问，即便 Android 有沙盒机制，仍然可以轻易突破[5]。Android 上的权限提升攻击主要集中在没有认证的语音、文本通信、一些恶意代码、上下文感知的话音记录的下载等，比较典型的例子就是 Android 上的 Opera Mobile[6] 不安全文件权限缓存“毒药”漏洞。Opera Mobile 是一款用于在智能手机和个人数码助理上浏览网页的网络浏览器，它的缓存文件（metadata 和 data）使用不安全文件权限：

1）缓存 metadata 文件（dcache4. url）全局可读和写。

2）缓存数据自身全局可读可写。

因此，没有正确权限的第三方应用程序可访问 Opera Mobile 缓存，突破 Android 沙盒模型[7]。攻击者可以利用此漏洞获得敏感信息或破坏 Web 缓存文件。

3 Android 权限动态防御机制设计

要实现 Android 权限的动态防御主要分为两个步骤：

1）选择监测对象，即选出可能被权限提升漏洞攻击的应用，这样做可以使得防御更具针对性，减少不必要的运算。

2）对该应用进行 IPC 动态监测，如果发现被监测应用受到权限提升攻击就向用户发出提醒。

3.1 监测对象选择

应用的权限设置包含在 AndroidManifest. xml 文件中[8]，通过对 AndroidManifest. xml 文件的分析，可以获得应用所申请的系统的权限以及应用自身组件的权限设定，下面给出一个 AndroidManifest. xml 文件的例子：

```
<activity android: name="AppMarketClientMain"
…
        <activity android: permission=""
Android: exported="true" >
          <intent-filter>
……
```

应用自定义的权限中，每个组件（Activity、Service、Broadcast Receiver、Content Provider）的权限都有相应的标签权限和 exported 标签。当 exported 值为 true 时，表明该组件能被其他应用访问，而当 exported 标签值为 false 时，表示该组件不能被其他应用所访问。exported 缺省值需要根据组件是否有 intent-filter 决定[9]（Content Provider 组件除外，该组件缺省值始终为 true）。当没有 intent-filter 时，缺省值为 false，反之，缺省值为 true。

当某应用被安装时，主要检测该应用的声明权限和 exported 值，如果声明的权限为敏感权限且 exported 值为 true，则该应用可能被其他应用恶意访问，故被列为监测对象，反之则不必监测。

3.2 IPC 动态监测

对于两个应用间的访问，整个权限保护的过程如图 3 所示。对于被列为监测对象的应用 B（默认 B 是受信任的安全应用），当它被非信任应用 A 访问时，通过调用 Android 的 IPC[10]，监测应用间通信的 Intent，可以获取是哪两个应用存在通信以及申请的权限。

设申请的权限为集合 $P=\{p_1, p_2\cdots\cdots\}$，应用 A 声明的权限为 $P'=\{p'_1, p'_1\cdots\cdots\}$，如果 $P\neq P'$，则表示应用 A 希望通过应用 B 进行其他权限的获取。将权限 $P_1=P-P\cap P'$ 即 P 中除去权限 A 声明的部分，与敏感权限集合 P_0 比较，看其中是否存在敏感权限。敏感权限 P_0 由使用者根据自身情况增加或减少权限，但会给出一个缺省的敏感权限集合 P_0，其中包括：android. permission. READ _ SMS（读取短信内容）、android. permission. READ_ CONTACTS（允许应用访问联系人通讯录信息）、android. permission. INTERNET（访问网络连接，可能产生 GPRS 流量）、android. permission. ACCESS_ FINE_ LOCATION（通过 GPS 的定位信息，定位精度达 10 米以内）[11]等。如果存在敏感权限，则可以判定应用 A 发起了权限提升攻击，如果没有敏感权限，则认为 A 访问 B 为安全访问，将 A 列入信任应用。

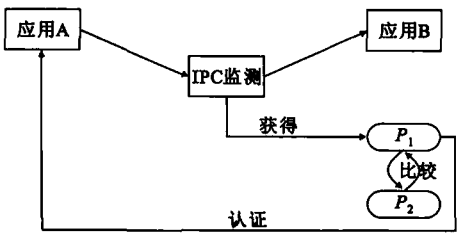

图 3 两进程权限保护的过程

4 实验验证

本次实验测试的环境为：某品牌手机，处理器双核 1.7 GHz，Android 版本为 4.1.1，内核为 Linux kernel 3.4.0。

实验设计应用如下：

1）短信管理。该应用为普通应用，主要用于将用户认为重要的短信存储起来作为备份，运行时字体大小为18 dp。

2）短信窃取。该应用为恶意应用，主要用于将短信管理应用中保存的短信窃取，运行时字体大小为25 dp。

3）权限提升检测。该应用为权限提升动态防御应用，主要用于检测权限提升是否发生，并提醒用户。

攻击实现如下：

1）运行短信管理应用，该应用首先声明一个 Cursorcur=cr. query（uri, projection, null, null. " date desc"）用于向系统申请读取短信，然后将读取的短信通过下面的遍历逐一显示：

```
do {
    name = cur.getString(nameColumn);
    phoneNumber = cur.getString(phoneNumberColumn);
    smsbody = cur.getString(smsbodyColumn);
    if (smsbody == null) smsbody = "";
    Map<String, Object> listItem = new HashMap<String, Object>();
    listItem.put("name", name);
    listItem.put("content", smsbody);
    listItems.add(listItem);
} while(cur.moveToNext());
```

运行结果如图 4 所示。

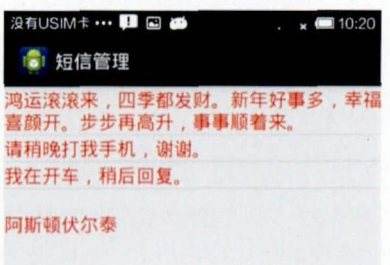

图 4 短信管理运行结果显示

2）运行短信窃取，首先建立一个 Intent intent = new Intent(); intent.setClassName("com.example.smsb","com.example.smsb.ReturnSms")，用于向系统申请访问应用短信管理，然后发送 Intent，获取结果，再将获得的数据包中的数据提取出来并显示，运行结果如图 5 所示。

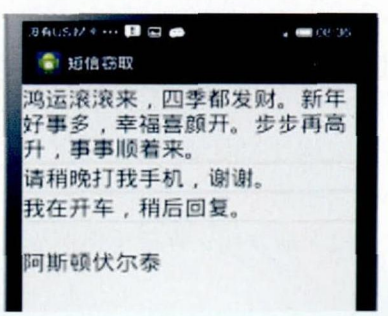

图 5 短信窃取运行结果显示

防御监测过程为：

1）监测对象选择。首先通过应用的 AndroidManifest.xml 文件，发现短信管理声明权限为 android.permission.READ_SMS，该权限表明这个应用可以读取短信，涉及到用户隐私，故该权限为敏感权限，同时 intent-filter 中 exported 的值为 true，所以将短信管理声明列为监测对象。

2）运行权限提升检测，输入监测应用为短信管理，通过动态防御的监测，发现短信窃取存在权限提升攻击的行为。当再次运行权限提升时，就会出现提示，告知使用者该应用存在权限提升攻击的恶意行为，运行结果如图 6 所示。

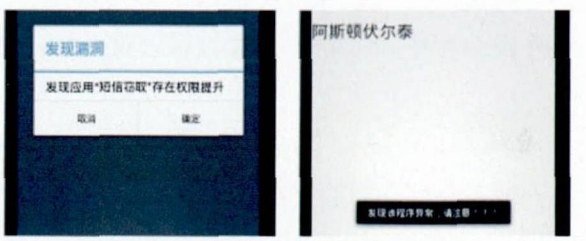

图 6 权限提升检测运行结果及提醒示意

6 结语

文章通过对 Android 平台上权限提升攻击的详细分析，对于常见的权限提升攻击的防御方法进行了总结和归纳，同时将 AndroidManifest.xml 文件分析和 Android 的 IPC 机制相结合，提出动态检测机制，并在 Android 平台上加以实验，证明可以有效地检测出权限提升攻击。当然，这种机制还存在一定的不足，主要有：

1）这种机制与用户的交互不够，当发生误判时，没有提供给用户修改结果的接口。

2）这种机制只能在权限提升发生以后提醒，并不能及时地制止权限攻击的发生，这些都将成为今后研究改进的方向。

参考文献：

[1] 蒋绍林，王金双，张涛，等. Android 安全研究综述 [J]. 计算机应用与软件，2012，29（10）：205-210.

（下转第 79 页）

Pointer Inference and JIT Spraying [R]. Black Hat Federal, 2010.

[2] CHEN Xiabo, XIE Jun. Defeat Windows 7 Browser Memory Protection [R]. XCon 2010.

[3] Derek Soeder. Memory Retrieval Vulnerabil-iYes [EB/OL]. http://www.eeye.com/eEyeDigitalSecurity/media/ResearchPapers/eeyeMRV-Oct2006.pdf, 2006.

[4] 张明磊, 单蓉胜, 李小勇. 基于 Windows 系统调用的异常检测模型 [J]. 信息安全与通信保密, 2007 (11): 25.

[5] Ben Hawkes. Attacking the Vista Heap [EB/OL]. http://www.lateralsecurity.com/downloads/hawkes_ruxcon-nov-2008.pdf, 2008-11.

[6] Chris Valasek. Understanding the Low Fragmentation Heap [EB/OL]. http://illmatics.com/Understanding_the_LFH_Slides.pdf, July, 2010.

[7] Tarjei Mandt. Kernel Attacks through User-Mode Callbacks [R]. USA, Black Hat, 2011.

[8] 羊建林, 周安民. Windows 异常处理与软件安全 [J]. 信息安全与通信保密, 2011 (4): 32.

[9] 马一楠, 张立和. Windows 下缓冲区溢出保护机制及绕过技术 [J]. 计算机工程, 2010, 36 (17): 147-151.

[10] Chris Valasek, Tarjei Mandt. Windows 8 Heap Internals [R]. USA, Black Hat, 2012.

[11] 赵婷婷, 陈小春, 杨娟. 基于 Windows 平台下的入侵检测系统 [J]. 通信技术, 2007 (12): 113.

[12] 潘爱民. Windows 内核原理与实现 [M]. 北京: 电子工业出版社, 2010. ■

(上接第 74 页)

[2] BARRERA D, KAYACIK H G, VAN Oorschot P C, et al. A Methodology for Empirical Analysis of Permission-based Security Models and Its Application to Android [C] //Proceedings of the 17th ACM Conference on Computer and Communications Security. ACM, 2010: 73-84.

[3] Vidas T, Christin N, Cranor L. Curbing android permission creep [C] //Proceedings of the Web. 2011, 2.

[4] DAVI L, DMITRIENKO A, SADEGHI A R, et al. Privilege Escalation Attacks on Android [M] //Information Security. Springer Berlin Heidelberg, 2011: 346-360.

[5] BUGIEL S, DAVI L, DMITRIENKO A, et al. Xmandroid: A New Android Evolution to Mitigate Privilege Escalation Attacks [J]. Technische Universit 8130Darmstadt, Technical Report TR-2011-04, 2011.

[6] BUGIEL S, DAVI L, DMITRIENKO A, et al. Towards Taming Privilege-escalation Attacks on Android [C] //Proceedings of the 19th Annual Symposium on Network and Distributed System Security. 2012.

[7] 杨博, 唐祝寿, 朱浩谨, 等. 基于静态数据流分析的 Android 应用权限检测方法 [J]. 计算机科学, 2012, 39 (z3): 16-18.

[8] 杨广亮, 龚晓锐, 姚刚, 等. 一个面向 Android 的隐私泄露检测系统 [J]. Computer Engineering, 2012, 38 (23): 1-6.

[9] FELT A P, CHIN E, HANNA S, et al. Android Permissions Demystified [C] //Proceedings of the 18th ACM Conference on Computer and Communications Security. ACM, 2011: 627-638.

[10] 戴威, 郑滔. 基于 Android 权限机制的动态隐私保护模型 [J]. 计算机应用研究, 2012, 29 (9): 3478-3482.

[11] ZHOU Y, ZHANG X, JIANG X, et al. Taming Information-stealing Smartphone Applications (on android) [M] //Trust and Trustworthy Computing. Springer Berlin Heidelberg, 2011: 93-107. ■