

Android 界面劫持攻击检测

王伟平¹, 高跃进, 林漫涛

(中南大学 信息科学与工程学院, 湖南 长沙 410083)

摘要: 针对于界面劫持攻击, 本文分析了造成界面劫持攻击的一般过程, 验证了这种攻击在多种版本的 Android 平台的有效性, 并提出了一种基于代码特征以及多组件数据流跟踪的静态检测方法 AHDetector (Activity Hijacking Detector)。AHDetector 通过关键代码定位后台扫描组件, 劫持界面组件以及隐私外传组件, 并进行数据流分析检测用户输入的隐私数据是否会泄露出去。实验结果表明, AHDetector 方法能够有效的检测出具有界面劫持攻击行为的恶意应用。

关键词: 界面劫持; 隐私泄露; 数据流跟踪

中图分类号: TP393.08

文献标志码: A

Detection of Activity Hijacking

Wang Weiping¹, Gao yuejin, Lin Mantao

(School of Information Sci. and Eng., Central South Univ., Hunan Changsha 410083)

Abstract: This paper analyzes the process of Activity hijacking attack and verifies the validity of such an attack in a variety of versions of the Android platform, and proposes AHDetector, a static detection method for Activity hijacking attack. AHDetector is based on code features analysis and multi-component data flow tracking. AHDetector finds out the scanning component, the hijacking component, and the privacy theft component by their code features, and tracks privacy data flow to detect whether it leak out or not. The experimental result show that AHDetector can effectively detect malicious applications with activity hijacking behavior.

Key words: hijacking; privacy leakage; data flow tracking

近几年随着移动互联网的高速发展以及智能终端的普及, Android 手机凭借其平台的开发性, 被广泛的用于智能终端领域, 根据著名的国际数据公司 (IDC) 公布的统计数据显示^[1], 2015 年第二季度 Android 智能手机市场占有率为 82.8%, 遥遥领先于其他操作系统平台的智能手机。

由于 Android 手机上面存储着大量与用户隐私相关的数据, 随着 Android 手机的蓬勃发展, 以窃取用户隐私数据为目的的恶意应用亦如雨后春笋般涌现, 大量的用户遭受过恶意攻击, 仅在 2015 年第一季度, 就有 707396 的用户遭受恶意应用 WifiPassword 窃取隐私^[2]。

界面劫持攻击是一种以窃取用户输入的隐私信息为目的的 Android 恶意攻击, 其过程一般为: 攻击程序会在后台不断扫描当

前设备上应用的运行信息, 一旦目标应用正在启动, 攻击程序立即启动伪造界面覆盖在目标应用之上, 诱骗用户输入敏感信息并将其外传。

在检测用户隐私泄露方面, Enck 等人实现了动态污点跟踪系统 TaintDroid^[3], 该系统修改了 Android 虚拟机的字节码解析器, 对隐私数据进行污点标记, 并实时跟踪应用程序中隐私数据的传播, 能够有效的检测和记录隐私数据离开终端设备的行为:

FlowDroid^[4]、AndroidLeaks^[5]、ScanDal^[6]以及 Apposcopy^[7]等 Android 恶意代码检测工具通过静态分析检测出应用中调用读写并对其进行数据流跟踪。但目前方法主要针对用户电话号码、通信记录等明确隐私数据的检测, 且静态分析方法只能检测单个组件内的隐私泄露问题。而由于界面劫持通常是

收稿日期: 2016-05-31

基金项目: 国家自然科学基金资助项目 (61672543), 长沙市移动互联网产业项目 (2015 年)

作者简介: 王伟平 (1969—), 女, 教授, 博导. 研究方向: 信息安全. E-mail: wpwang@csu.edu.cn

*通信联系人 E-mail: wpwang@csu.edu.cn

网络出版时间: 此位置必预留 10 个五号字空位 网络出版地址:

骗取用户输入隐私信息到劫持界面，进而通过相应输出通道外传，直接使用现有相关数据流跟踪方法无法直接检测，一方面是用用户界面输入信息很多，都作为隐私数据源将会导致太多误报，同时由于静态分析方法主要针对单组件内部数据流进行检测，一旦界面劫持应用了多个组件，例如骗取用户输入界面与数据外传通道在不同的组件中，就无法直接进行检测。

而针对界面劫持攻击，文献[8，9]提出了利用图形相似性检测的方法，通过截取被检测应用与目标应用界面图像，并计算这两个图像的相似度从而判断是否存在界面劫持攻击，但是这种检测方法需要事先知道被检测应用伪造了哪个目标应用，并且没有对被检测应用行为与目的做进一步检测。

本文完成的主要工作和贡献包括：

- (1) 分析了界面劫持攻击的过程，总结了实现劫持攻击的方式，开发了相关的界面劫持样例代码验证了界面劫持攻击在多种版本的安卓系统上都能成功实施。
- (2) 设计并实现了一种基于代码特征分析以及多组件数据流跟踪的静态检测方法 AHDetector（Activity Hijacking Detector）。该方法通过检测扫描组件及其启动的组件来定位需检测的输入界面；同时设计了一种跨组件的数据流跟踪方法，通过跨组件的数据流分析方法检测界面输入到外传通道是否有数据流传递。
- (3) 设计了12种覆盖完整的界面劫持样例以及6种容易误判的样例，同时将4种具有扫描拦截功能的应用锁也作为容易混淆的样例，测试结果表明 AHDetector 能够有效地检测出具有界面劫持攻击行为的应用，同时没有出现误判。

1 界面劫持攻击分析和测试

1.1 公式

首先我们对界面劫持行为进行抽象，它主要包含三个功能组件：后台扫描组件、劫

持界面组件、隐私数据组件，如图1所示，它们的主要行为表现如下：

- (1) 后台扫描组件，负责启动一个后台不断扫描的服务，来获取当前设备上应用的信息，一旦扫描到当前用户启动的界面为被劫持界面，则启动精心构造的劫持界面；
- (2) 劫持界面组件，针对于特定应用伪造的输入界面，诱骗用户输入敏感信息；
- (3) 隐私外传组件，将从劫持界面中获得的用户输入通过短信，网络等方式外传。

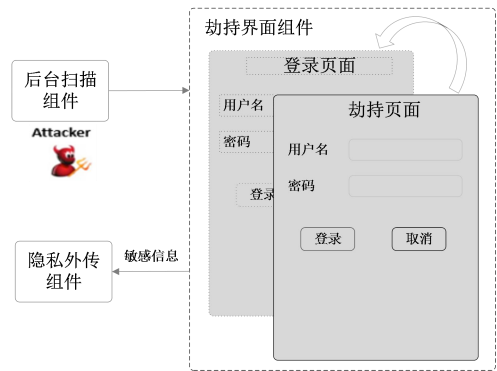


图1 界面劫持攻击示例
Fig.1 Activity Hijacking Sample

1.2 攻击测试

为了研究界面劫持攻击影响的 Android 系统版本范围，我们实现了一个界面劫持攻击程序，该程序针对手机 QQ 登录界面进行劫持。将该程序在 Android 不同系统版本的模拟器和真机上面进行了测试。测试结果如表1所示，在 Android 2.3-4.4 版本的测试平台上该攻击程序都能够成功劫持，而在 Android 5.0 以上版本的测试平台上，攻击程序未能成功劫持。我们通过查阅 Android 开发文档，发现 Android 5.0 以上的系统对获取当前程序运行信息相关 API 的调用做了严格的限制，普通的应用程序通过后台扫描只能获取自己的运行信息，而无法获取其它程序的运行信息，这增加了针对特定界面启动时的攻击难度，但劫持代码依然可以弹出欺骗性的界面。

本文还测试了国内主要的 Android 安全软件（360 安全卫士、腾讯手机管家、百度

手机卫士、LBE 安全大师)对界面劫持攻击的检测效果,很遗憾,这些安全软件目前并不能对界面劫持攻击进行检测与防护。

通过上文的分析,可以看到攻击程序在 Android 5.0 版本以下的手机能够在劫持目标启动时,完成攻击。界面劫持攻击具有较大的危害性,具体表现在以下四个方面:①针对性强,攻击者可以劫持任何他感兴趣的应用界面,比如涉及账户登录、支付以及填写个人重要信息的应用界面;②成功率高,目前还没有安全软件做防护与检测,并且攻击行为隐蔽,普通用户一般难以察觉;③后果严重,受到界面劫持攻击的用户可能会无法使用目标应用,泄露个人信息,甚至遭受财产损失;④影响范围广,目前攻击程序在主流 Android 系统版本(2.3-4.4)的手机上都能运行。

表 1 不同平台测试结果

Tab.1 Test result for different platforms		
测试平台	Android 系 统版本	是否 成功
Xperia Pro	2.3	√
vivo xplay	4.2	√
huawei p6-T100	4.3	√
nubia z9	5.0	×
Android 模拟器	2.3-4.4	√
Android 模拟器	5.0-5.1	×

2 AHDetector检测思想与方案设计

2.1 界面劫持攻击的必要条件

根据界面劫持攻击的分析可知,实现界面劫持攻击的必要条件是:

(1) 具有外传数据的敏感权限,由于界面劫持攻击最终目的是为了窃取用户的隐私数据并外传,所以恶意软件必须具有外传数据的敏感权限,如:发送短信、连接网络、读写文件等;

(2) 同时具备后台扫描组件、劫持界面组件、隐私外传组件;

(3) 后台扫描组件、劫持界面组件之间有先后的调用关系;

(4) 劫持界面组件和隐私外传组件之间有数据流通路,并将劫持界面组件中用户

的输入外传。

2.2 AHDetector的检测思想与设计

根据界面劫持攻击的必要条件,本文提出了一种针对于界面劫持攻击的检测方法 AHDetector(Activity Hijacking Detector),图 2 展示了 AHDetector 的总体设计思想,主要分为以下几步:

(1) 分析被检测应用申请的敏感权限,判断是否具有外传数据的敏感权限,如果有则进行下一步的分析,否则结束分析,判定该应用没有界面劫持攻击行为。

(2) 根据后台扫描组件、劫持界面组件、隐私外传组件的代码特征,判断被检测应用中是否同时存在这三种组件的疑似组件,即同时具有扫描功能的组件、接受界面输入的组件和具有外传数据通道的组件。如果有,则进行下一步分析,否则结束分析,判定该应用不包含界面劫持攻击;

(3) 通过分析组件间调用关系,判断具有扫描功能的组件与接受界面输入的组件之间是否存在调用关系,如果有,则进行下一步分析,否则结束分析,判定该应用不包含界面劫持攻击;

(4) 通过组件间数据流分析,确定劫持界面组件和隐私外传组件之间是否存在隐私数据的传递,即是否存在将界面输入数据传递到具有外传数据通道的组件。如果有,则判定该应用具有界面劫持攻击,否则没有。

3 AHDetector实现

3.1 敏感权限检测

每一个 Android 应用程序都包含一个 AndroidManifest.xml 的配置文件,该配置文件提供了程序运行的必要信息,其中包含了敏感权限的申请。AHDetector 通过分析应用的配置文件,检测应用是否具备表 2 所示的外传数据敏感权限,包括短信外传通道、外部存储通道和网络访问通道。具体实现的方法是利用 Androguard 自身提供的 dad 反编译器^[10]反编译被检测应用,提取出 AndroidManifest.xml 文件,通过字符串匹配判断该文件中是否存在外传数据敏感权限

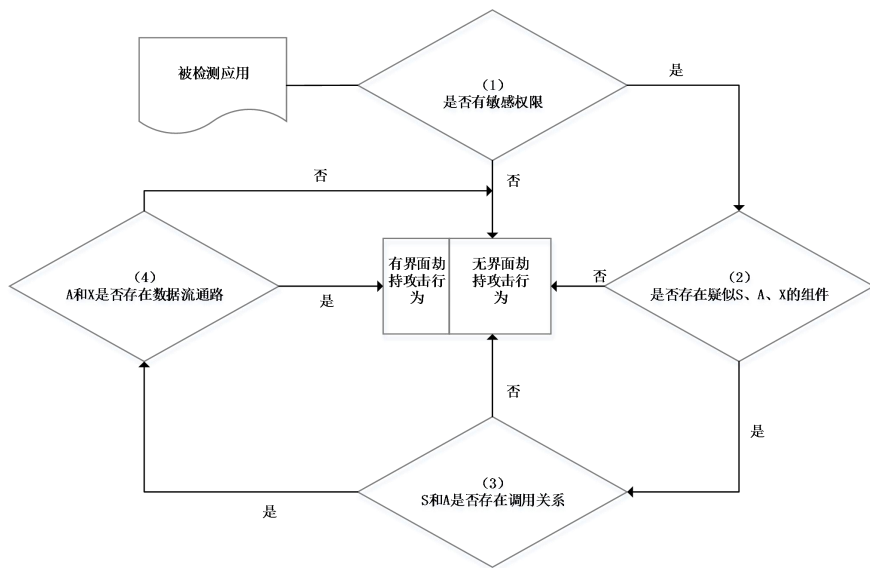


图 2 AHDetector 总体设计（其中 S 为后台扫描组件；A 为劫持界面组件；X 为隐私外传组件）

Fig.2 Overall design of AHDetector(S:scanning component;A:hijacking component;X: privacy theft component)

列表中任意一个权限。

表 2 外传数据敏感权限

Tab.2 Sensitive permissions to transmit data	
权限名称	权限作用
发送短信	android.permission.SEND_SMS
编写短信	android.permission.WRITE_SMS
写入外部存储	android.permission.WRITE_EXTERNAL_STORAGE
访问网络	android.permission.INTERNET

3.2 判断是否同时存在三种疑似组件

由于界面劫持攻击包含的三种功能组件特征显著，在实现上会调用特定的系统 API，如后台扫描组件会调用 `getRunningAppProcesses` 来获得当前设备正在运行的进程列表进而判定被劫持的页面是否调用。

AHDetector 根据这三个功能组件的代码特征来判定可疑组件，表 3 给出了用于判定这三种疑似组件的关键 API 示例。

3.3 调用关系判定

为了判断具有扫描的组件是否会调用接受界面输入组件，我们首先分析应用各组件之间的调用方式。

在 Android 应用中，组件之间的相互调用是通过组件间通信方法来实现的，这些组

表 3 功能组件关键 API 示例

Tab.3 API examples of function components

疑似组件	关键 API	功能描述
具有扫描功能的组件	<code>getRunningTasks</code>	返回当前设备运行的任务栈
	<code>getRunningAppProcesses</code>	返回当前设备正在运行的进程列表
	<code>logcat</code>	执行 logcat 命令，读取当前设备的运行信息
接受界面输入的组件	<code>findViewById+EditText+getText</code>	获取输入控件的内容
具有外传数据通道的组件	<code>sendTextMessage</code>	发送短信
	<code>openConnection</code>	联网操作

件间通信方法可以通过显式或者隐式的 intent 来指定要调用的目的组件。

3.3.1 Android 应用组件调用方法

为了明确 Android 应用中组件是如何相互调用的，图 3 展示了组件间调用的例子，图中虚线上方展示了组件 A 显式调用组件 B 的过程，虚线下方展示了组件 C 隐式调用组件 D 的过程。

显式的 intent 明确指定要调用的组件类名，如图 3 中 A 通过调用 intent 的 `setClass` 方法来指定要调用的组件类名 B。

隐式的 intent 没有明确指定要调用的组件，只是设置 intent 需要调用的组件应满足的条件。任一组件可以通过 IntentFilter 来声明能够处理哪些条件下的隐式 Intent。当一个组件的 Intent 设置的条件与另一组件的 IntentFilter 声明的条件匹配时，该组件就可以调用另一个组件。如图 3 中组件 C 隐式调用组件 D 的过程，具体的方法是组件 C 通过 intent 的 setAction 方法指定了要调用组件应满足 action 值为“myaction”的条件。组件 D 在其 AndroidManifest.xml 文件中通过 IntentFilter 声明的 action 值为“myaction”，因此组件 D 满足组件 C 要调用的目的组件的条件，系统就会调用组件 D。

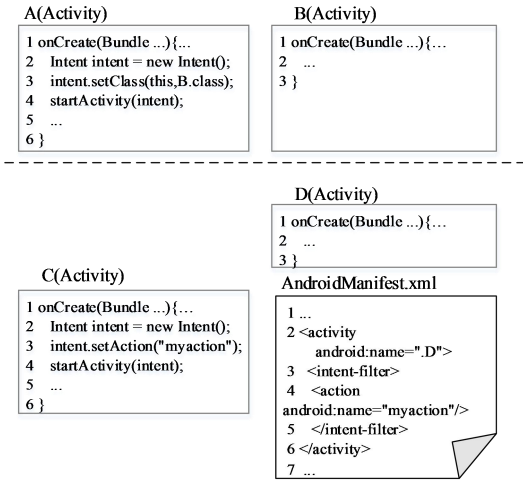


图 3 组件间调用关系示例

Fig.3 Example of call relationship between components

3.3.2 AHDetector 检测组件调用关系的方法

AHDetector 分析每一个组件中的组件间通信方法和 Intent 对象，确定其调用的组件。

首先判定该组件的显式调用 Intent，通过解析其 setClass 方法或者构造方法的参数来确定要调用的目的组件。

接着判定隐式 Intent，通过查找 Intent 关键方法调用，比如 setAction、addCategory，或是分析 Intent 的构造方法，记录下其中 action、category 以及 data 等属性值。由于隐式调用中，符合该属性的组件都可以成为被调用组件。需要分析 AndroidManifest.xml 文件中所有组件的 IntentFilter 标签下所定义

的 action、category 以及 data 等属性值，将所有匹配的组件定为调用的目的组件。

AHDetector 通过遍历每一个组件，在调用组件之间增加调用边，就得到应用所有组件之间的调用关系有向图。在该图中，判定具有扫描功能的组件与接受界面输入的组件之间是否存在路径，若有，则认为两者之间存在调用关系。

3.4 组件间数据流分析

为了确定从劫持界面组件获取的用户输入是否被外传，AHDetector 将界面劫持攻击的检测转化为跨组件的数据流分析，即判定界面组件是否将界面输入信息传递给具有外传数据通道的组件并将该信息传递出去。具体实现的方法是：首先对每个组件进行组件内数据流跟踪，判定组件在隐私数据传递通路中可能的角色，然后结合组件间调用关系，判断隐私数据的传递通路。

3.4.1 组件内数据流分析

组件内数据流分析的目标是确定组件与隐私数据传递通路可能的关系。如图 4 所示，当一个组件位于隐私数据传输通路上时，可能充当的角色有四种类型：组件直接获取隐私数据并将其直接通过外传通道发送出去，用 r 表示；组件直接获取隐私数据并将其传入到其它组件中去，用 y 表示；从其它组件获取数据并将其传入到其它组件中去，用 g 表示；从其它组件获取数据并将其通过外传通道直接发送出去，用 b 表示。

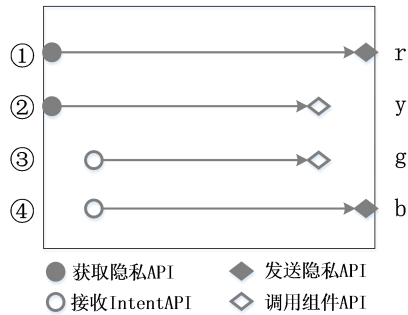


图 4 组件内部隐私传递

Fig.4 Privacy transfer within a component

本文利用开源的 Android 组件内静态污点分析工具 FlowDroid[4]进行组件内的数据流分析。由于 FlowDroid 的分析结果是每一个组件内部存在的污点数据流，并不能将组

件按照图 4 中显示的四种组件类型进行分类。为了判定组件的类型，我们首先将 Flowdroid 的 source 和 sink 节点 API 分为两类：（1）直接与隐私相关的 source 和 sink；（2）组件间通信相关的 source 和 sink，如表 4 所示。

表 4 source 和 sink 分类

Tab.4 Classification of source and sink

	隐私相关 API	组件间通信相关 API
source	直接调用获取隐私数据的 API 如 <code>getText</code>	由其它组件通过 <code>Intent</code> 对象传入隐私, 如 <code>getIntent</code>
sink	直接调用调用发送隐私数据的 API 如 <code>sendTextMessage</code>	启动并将隐私数据传给其它组件, 如 <code>startActivity</code>

根据不同类型 source 和 sink 的组合，得到图 4 中的四种组件分类。然后根据 FlowDroid 得到的每一个组件内的污点数据流，判定 source 和 sink 的组合，确定组件的类型是属于 r、y、g、b 中的哪一种，并依次存放到相应的集合 R、Y、G、B 中。

3.4.2 组件间数据流分析

组件间的数据流分析的目标是确定界面输入的隐私数据的传递通路。Android 中组件间是通过 Intent 传递数据的，如图 5 展示了组件间递数据的示例，组件 A 获取隐私数据存储在变量 info 中，通过 Intent 以键值对<"data", info>的形式将 info 传递给组件 B，组件 B 根据隐私数据的键“data”获取隐私数据。因此我们通过比较发送组件传递的 Intent 和接受组件获取的 Intent 携带数据的键值是否一致，判定组件之间是否存在数据的传递。

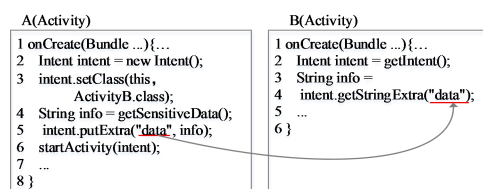


图 5 组件间数据传递示例

Fig.5 Example of data transfer between components

组件间的数据流分析主要包含如下几步:

- (1) 依据组件内数据流分析的结果,

标注组件调用有向图中各个组件属于 R、Y、G、B 哪种集合；

(2) 组件调用有向图中, 去除不属于四种类型集合中的组件及其与之相连的边, 得到组件调用有向图的子图:

(3) 从 Y 集合组件开始, 对子图进行深度遍历;

(4) 在遍历的过程中, 需要对边两边的组件传递的 **Intent** 和获取的 **Intent** 携带的数据的键值进行分析, 如果两个键值是一致的, 则继续深度遍历, 否则, 不考察此边, 回退到上一节点继续深度遍历;

(5) 对于每个 Y 集合组件, 分别找出其与 B 集合组件的路径, 有路径则表明这两个组件有隐私传递, 否则没有隐私传递。

图 6 展示了组件间数据流分析的示例, 假定被检测应用有 7 个组件。对该应用进行组件间的数据流分析主要包含如下几个步骤:

(1) 图 6 (a) 是得到的组件调用有向图, 依据组件内的数据流分析的结果对组件调用有向图中的组件进行标注, 如图 6 (b) 所示;

(2) 将组件调用有向图中不属于四种集合的组件以及与之相连的边去掉, 得到图 6(c) 所示的子图;

(3) 以节点 4 为起点, 遍历类型为 g 或 b 的节点, 如图 6(d) 所示, 找到节点 5, 分析得到节点 4 和 5 之间的 Intent 键值对匹配, 继续进行遍历, 假设节点 5 和节点 7 之间的 Intent 键值对不匹配, 抛弃这条边, 如图 6(e) 所示;

(4) 进行回溯, 如图 6 (f) 所示找到节点 6, 分析得到节点 4 和节点 6 之间的 Intent 键值对匹配, 继续遍历找到节点 7, 分析得到节点 6 和节点 7 之间的 Intent 键值对匹配, 由于节点 7 的类型为 b, 停止遍历, 得到节点 4 到节点 7 之间存在一条传递数据的路径, 如图 6 (g) 所示。

AHDetector 通过对被检测应用进行组件间的数据流分析, 确定劫持界面组件和隐私外传组件之间是否存在隐私数据的传递, 即是否存在将界面输入数据传递到具有外传数据通道的组件, 若有, 则判定被检测应

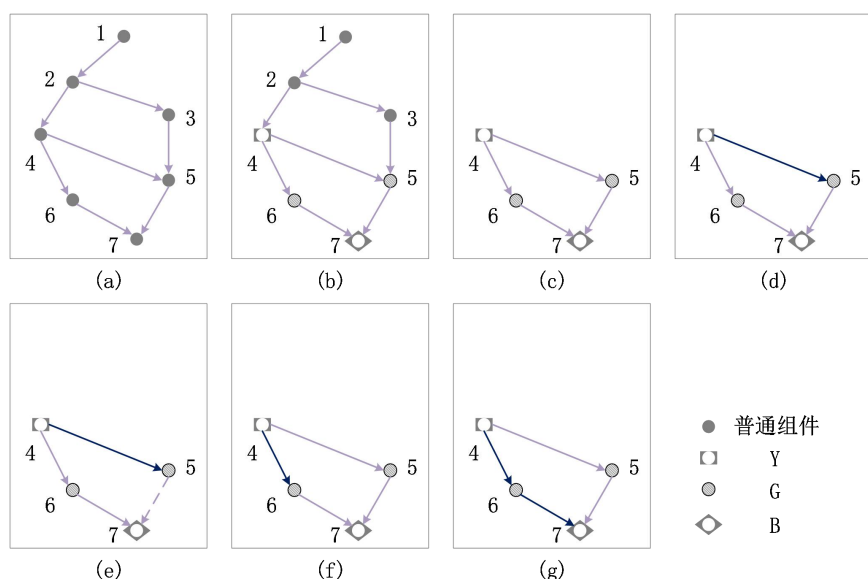


图 6 组件间数据流分析示例

Fig.6 Example of data flow analysis between components

用具有界面劫持攻击，否则没有。

表 5 测试样例

Tab.5 Test sample

4 测试评估

界面劫持攻击的三个功能组件可以通过直接或间接调用，并且劫持界面组件最终会将用户输入的信息经由隐私外传组件泄露出去，如图 7 所示。

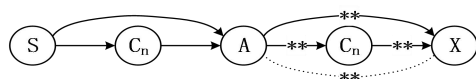


图 7 界面劫持功能组件逻辑路径(其中 S 为后台扫描组件；Cn 为 1 个到多个组件；A 为劫持界面组件；X 为隐私外传组件；**表示有隐私数据传递；虚线表示 A 与 X 在同一个组件中实现)

Fig.7 Logical path of function

components(S:scanning component;Cn:1 or more components;A: hijacking component;**:privacy data transfer;dotted line:A and X are in same component)

为了测试 AHDetector 检测效果，本文依据界面劫持攻击功能组件的逻辑关系，设计了 18 个测试样例，包括两大类：有界面劫持攻击样例和无界面劫持攻击样例。

此外，在 Android 应用市场中存在着一类名为应用锁的程序。安装了应用锁的用户可以自己设定要锁定的应用程序，当目标程序启动时，应用锁就会弹出界面要求用户输入信息，如解锁密码。应用锁与界面劫持攻

分类	编号	描述
S 与 A	01	S 调用 A
路径关	02	S 调用 C _n ，C _n 调用 A
系		
A 与 X	03	A 调用 X
有	04	A 调用 C _n ，C _n 调用 X
路径关		
系	05	A 与 X 在同一个组件中实现
面	06	S 通过 getRunningAppProcesses
劫	S 扫描	扫描
持	方式	07 S 通过 getRunningTasks 扫描
攻	08	S 通过读取 Logcat 日志扫描
击	09	X 通过短信发送隐私
X 隐私	10	X 通过网络发送隐私
方法方	11	X 将隐私写到 SD 卡上
式	12	X 将隐私写到自定义的 ContentProvider 中
缺少组	13	不存在 A
无	14	不存在 S
件	15	不存在 X
面	16	S 没有调用 A
无	17	A 没有调用 X
路径		
劫	18	A 没有发送隐私到 X
关系		
持	19	com.***.appslocker
攻	20	com.***.applocker
击	21	com.***.applock
应用锁	22	com.***.locker

表 6 测试结果

Tab.6 Test result

样例 编号	Andrubis	VirusTotal	visualThreat	安全管家在 线测	腾讯安全实 验室	网秦安全	AHDDetect or
01	-	-	-	-	-	-	√
02	-	-	-	-	-	-	√
03	-	-	-	-	-	-	√
04	-	-	-	-	-	-	√
05	-	-	-	-	-	-	√
06	-	-	-	-	-	-	√
07	-	-	-	-	-	-	√
09	-	木马	-	-	-	-	√
10	-	-	-	-	-	-	√
11	-	-	-	-	-	-	√
12	-	-	-	-	-	-	√
13	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-
16	-	-	-	-	-	-	-
17	-	-	-	-	-	-	-
18	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	-	-	-	-	-	-	-
21	-	-	-	-	-	-	-
22	-	-	-	-	-	-	×

注：“√”表示检测出攻击行为；“-”表示未检测出攻击行为；“×”表示无法分析测试样例；其它信息表示检测出其它恶意行为。

击程序类似，都会在目标应用启动时弹出其它界面，所以本文也将其加入到测试样例中，在进行测试前，本文先对应用锁类的 4 个样例进行人工分析，确定了这 4 个样例皆无界面劫持攻击行为。

这 22 个测试样例的简要描述如表 5 所示，其中 S、A、X 分别代表界面劫持中的后台扫描、劫持界面以及隐私外传组件，Cn 表示它们之间经过的 n 个组件，应用锁类样例给出了部分包名信息。

在对 AHDDetect 进行测试的同时，本文还在 6 个常见的 Android 恶意应用在线检测平台提交了表 5 描述的测试样例，这些检测平台包括 Andrubis、VirusTotal、visualThreat、安全管家在线测、腾讯安全实验室以及网秦安全，测试结果如表 6 所示。

01 号-12 号样例为存在界面劫持攻击的

样例，AHDDetect 能够全部检测出其攻击行为，而其它检测平台不能检测出攻击行为，并且 VirusTotal 还将 09 号样例检测为木马程序，它是唯一调用发送短信 API 的测试样例，所以 VirusTota 可能将发送短信的行为作为检测木马程序的特征。

13 号-18 号样例是我们设计的没有界面劫持攻击行为的样例，19-22 是密码锁软件，由于该类软件具备扫描组件，我们也将这四个软件列为测试对象，测试表明 AHDDetect 不存在误报。只是 AHDDetect 分析 22 号样例时，分析失败了，这主要是 AHDDetect 在组件内数据流跟踪时用到了 FlowDroid，FlowDroid 静态分析还有些不足，特别是对于复杂应用，FlowDroid 运算过程会消耗过多的资源而导致分析失败。

5 总结

本文针对界面劫持攻击,分析了其攻击的过程,总结了实现劫持攻击的原因,提出了一种基于代码特征以及多组件数据流跟踪的静态检测方法 AHDetector。AHDetector 有效的将界面劫持攻击的检测问题转化为数据流分析的问题,实现了多组件的数据流跟踪。测试结果表明 AHDetector 能够有效的检测出应用是否具有界面劫持攻击的行为。

参考文献:

- [1]IDC.Smartphone OS Market Share,2015 Q2[EB/OL].IDC,[2016-03-01].<http://www.idc.com/pr odserv/smartphone-os-market-share.jsp>.
- [2]360 互联网安全中心.2015 年中国手机安全状况报告 [EB/OL].[2016-01-29].<http://zt.360.cn/1101061855.php?dtid=1101061451&did=1101593997>.
- [3]Enck W,Gilbert P,Han S,et al. TaintDroid:an information-flow tracking system for realtime privacy monitoring on smartphones[J].ACM Transactions on Computer Systems (TOCS),2014,32(2):5.
- [4]Arzt S,Rasthofer S,Fritz C,et al. Flowdroid: Precise context,flow,field,object-sensitive and lifecycle-aware taint analysis for android apps[C]//ACM SIGPLAN Notices.ACM,2014,49(6):259-269.
- [5]Gibler C,Crussell J,Erickson J,et al. AndroidLeaks:automatically detecting potential privacy leaks in android applications on a large scale[M]. Springer Berlin Heidelberg,2012.
- [6]Kim J,Yoon Y,Yi K,et al. ScanDal: Static analyzer for detecting privacy leaks in android applications[J].MoST,2012,12.
- [7]Feng Y,Anand S,Dillig I,et al. Apposcopy: Semantics-based detection of android malware through static analysis[C]//Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering.ACM,2014:576-587.
- [8]Liu YONG-Ming,Yang Jing.Detection of Android Phishing Malwares Based on Image Similarity[J].Computer Systems & Applications,2014,23(12):170-175.[刘永明, 杨婧.基于图像相似性的 Android 钓鱼恶意应用检测方法

[J].计算机系统应用,2014,23(12):170-175.]

- [9]Xu Qiang,Liang Bin,You Wei,et al.Detecting Android Malware Phishing Login Interface Based on SURF Algorithm[J].Jotunal if Tsinghua University(Science and Technology),2016,(01):77-82.[徐强,梁彬,游伟,等.基于 SURF 心算法的 Android 恶意应用钓鱼登录界面检测 [J].清华大学学报(自然科学版),2016,(01):77-82.]
- [10]androguard.Androguard[EB/OL].androguard,2013-06-30[2016-03-01].<http://androguard.blogspot.de/>.