3.3 Android应用漏洞检测模块

在3.2所述的恶意应用鉴别模块判别出是否为恶意应用后，紧接着运行的是静态检测中的第二个模块——漏洞检测模块。Android应用中或多或少都可能存在一些编写上存在漏洞的代码，这些Android应用漏洞虽然可能并不含有恶意代码，无法直接被3.2的恶意应用鉴别模块检测出来，但在被攻击者恶意利用的情况下，同样可以对用户产生相当严重的威胁，降低应用的安全性。基于上述情况，本模型从静态分析角度入手，设计了Android应用漏洞检测模块。

传统的检测方案是利用反编译工具将dex文件转换为smali中间代码，之后对漏洞特征的匹配都是在smail代码上基于字符匹配完成的，这种方式虽然简单，但无法对需要结合上下文语义、逻辑复杂的漏洞实现高精度的识别，因此本模块采用Soot进行反编译并在初始化中执行一些额外的工作。另一方面，为了提高漏洞检测精度，本文提出了一种基于过程内数据流分析技术的变量行为记录算法，在漏洞检测中可以用于实现函数内变量到相关常量的追溯，提高检测准确率。

3.3.1 Android应用漏洞检测模块的工作流程

漏洞检测模块的完整工作流程如图7所示。首先使用Soot对APK进行初始化工作，输出的两个结果一个是Jimple中间代码，另一个则是控制流图，其中控制流图使用3.3.2小节中介绍的追溯算法为函数内的变量生成行为记录。初始化工作结束后，则开始进行漏洞特征匹配，最后输出检测结果。



图7 漏洞检测模块的工作流程

3.3.2 基于数据流分析的变量行为记录算法

在对漏洞特征进行匹配时，经常将特定的常量参数做为漏洞特征，然而应用开发中有很多情况是先将常量赋值到一个变量对象里，进行处理之后再作为参数传值到漏洞特征函数中，一个最简单的例子就是使用StringBuilder连接字符串，最后调用toString()方法将生成的字符串作为参数传给函数，对于这种情况，传统基于字符匹配的检测方案是无法匹配到特征的，本小节提出的算法用于则可以解决这种问题，算法基于过程内数据流分析生成变量的行为记录，之后基于此记录追溯变量的常量特征。具体算法流程如图8所示



图8 变量行为记录算法

在利用Soot生成控制流图后，使用Soot提供的过程内数据流分析框架来模拟完整控制结构下的语句遍历分析，并在此过程中基于传入的记录规则，详细记录所有变量的初始化、变量的方法和成员变量调用、变量赋值、变量值传递等所有变量相关的行为，力求在函数内为所有局部变量的行为建立准确详细的模型。表3是过程内数据流分析框架中需要实现的两个主要接口，用于指定变量行为记录规则。

表3 过程内数据流分析框架的两个主要接口

|  |  |
| --- | --- |
| 方法名 | 作用 |
| flowThrough(d.inFlow,d.data,d.outFlow) | 对传入语句d.data进行语法分析，在d.inFlow参数的记录基础上，添加新的记录并作为d.outFlow传出 |
| merge(in1,in2,out) | 对分支结构两条路径的记录结果in1、in2进行合并，生成整合的记录结果out传出 |

flowThrough(d.inFlow, d.data, d.outFlow)函数的具体工作是对传入语句d.data进行分析，并添加新的记录，系统实现中根据不同的语句类型进行相应的解析来生成新的记录，几种常见的语句类型对应的操作如下：

1) IdentityStmt为变量初始化语句，对左边的被初始化变量以及右边的初始化类型进行记录。

2) DefinitionStmt为定义类型语句，左操作数是变量的话，根据右边的表达式种类，可能的操作有：使用工厂函数进行变量初始化、变量赋值变量方法调用操作，根据右侧表达式类型进行分析记录。

3) InvokeStmt为函数调用类型语句，里面可能存在变量的方法调用，将调用的方法解析出来并存放到对应变量的调用记录中去。

merge(in1,in2,out)函数用于将分支结构中两条路径的记录进行合并以整合成一条记录，实现中为了保证所有分支路径下变量调用记录的完整性，对两条记录做并集来生成整合记录，即。

通过如上定义的数据流分析规则，即可生成函数内变量的行为记录，然后在如下位置提取常量作为变量的常量特征：

1) 变量的函数调用中所应用的常量参数，如果参数为变量的话则进行递归查询。

2) 变量的定义处涉及到的常量，若定义源头来自另一个变量的话，则对此变量进行递归查询

3.3.3 以WebView远程代码执行漏洞的检测为例介绍漏洞检测过程

WebView远程代码执行漏洞[7]是Android应用中相当常见的一个漏洞，其危害也相当广泛，本文将以此为例介绍使用Soot的初始化，并结合变量行为记录算法下的WebView远程代码执行漏洞的检测方案。

检测的第一步使用Soot进行初始化工作，一方面进行反编译，另一方面则为应用构造控制流图。初始化分析结束后，需要提取出应用所使用的API版本号，对于API<17的应用，使用3.3.2中说明的算法生成函数内所有变量的行为集合，并在集合中分别寻找WebView类型以及WebViewSetting类型的变量，并在行为集合中定位到以下三个成员函数的调用记录

1) setJavaScriptEnabled()

2) removeJavascriptInterface()

3) addJavascriptInterface()

没有调用setJavaScriptEnabled()来开启JavaScript调用的WebView变量不存在此漏洞，而对于将JavaScript调用设置为允许的WebView，则需要进行进一步的检查。定位removeJavascriptInterface()函数的参数，如果参数为变量，则使用3.3.2中介绍的常量追溯算法将其转化为常量，之后检查参数是否为searchBoxJavaBridge，如果不是的话则说明没有移除默认接口，可判定为存在WebView远程代码执行漏洞。此外，如果移除了默认接口但又添加了新Java实例的WebView同样存在问题。

图 API<17时的WebView漏洞检测方案

对于API≥17的应用，因为提供了注解机制，所以直接查找addJavascriptInterface()函数的调用位置，之后从其中提取出传入的Java实例对象，并对此Java对象中使用JavaScript注解的成员方法进行分析，如果方法内含有危险API的话，同样判定此WebView存在漏洞

.

图 API≥17下的WebView漏洞检测方案

3.4 隐私泄露检测模块

在智能设备大量普及的当下，手机支付、手机社交都与移动设备紧密相关，可以说手机比电脑中存放着更多用户的隐私信息，这些信息一旦被应用恶意获取，就可能给用户带来无法挽回的损失。因此用户隐私安全性与Android应用的安全性是紧密相关的，隐私泄露检测模块便从隐私泄露的角度入手，进行Android应用安全性的评估

在下面的3.4.1小节中，首先对Android系统中涉及到用户隐私的API（即污点分析算法中的Sources点，以下简称Sources点），以及可能发送出这些隐私数据的API（即污点分析算法中的Sinks点，以下简称Sinks点）进行详细的分析和归类。在3.4.2小节中，介绍基于过程间数据流分析技术[8]，寻找从Sources点到Sinks点的传播路径的污点分析算法。在最后的3.4.3小节中，则展示本模块在批量测试中，对应用隐私泄露问题的检测效果。

3.4.1 对Sources点及Sinks点的分析归类

Android系统中含有很多涉及到获取用户隐私的API，例如能返回用户设别MEID的getDeviceId()、获取用户精确位置的getLastKnownLocation()、获取浏览器书签信息的的getAllBookmarks()、获取浏览器历史记录的信息getAllVisitedUrls()等，通过归类整理本文将其大致分为以下6类，如表4所示

表4 对Android中涉及用户隐私API的分类

|  |  |
| --- | --- |
| 隐私类别 | Android系统中相关API条目数 |
| 用户位置信息 | 7 |
| 用户唯一标志信息 | 6 |
| 用户私人数据 | 9 |
| 用户网络访问相关信息 | 10 |
| 应用组件信息 | 9 |
| 程序运行间数据信息 | 98 |

在恶意应用获取到用户隐私数据之后，同样可以使用多种手段将数据发送到攻击者手中，例如通过sendTextMessage()发送短信、通过openConnection()的HTTP请求发送数据、通过Log.d()以日志形式记录数据、通过FileOutputStream.write()将数据写入到本地文件等，在进行归类后我们将其大致分为以下5类

表5 对Android中数据发送相关API的分类

|  |  |
| --- | --- |
| 数据发送方式类别 | Android系统中相关API条目数 |
| 通过短信发送数据 | 3 |
| 通过网络请求发送数据 | 13 |
| 通过日志记录数据 | 14 |
| 通过本地文件记录数据 | 21 |
| 通过应用组件传递数据 | 108 |

基于对Sources点和Sinks点的归类整理，之后便可以使用基于过程间数据流分析技术的污点传播算法，寻找从Sources点到Sinks点的传播路径，从而发现这些隐私泄露问题。

3.4.2 使用污点分析算法寻找传播路径

Android应用程序的运行方式不像其他程序一样有个固定的入口点，然后顺序执行下去，Android应用中每一个组件都有自己完整的生命周期，组件在运行过程中随时可能被触发调用，因此要对Android应用程序进行过程间数据流分析，就需要先为Android应用程序的生命周期建模，并构造一个虚拟的main方法，在方法内模拟触发所有组件，然后再对这个虚拟的main方法进行数据流分析，寻找Sources到Sinks间的路径。

模型使用FlowDroid[9]进行实现污点传播路径的寻找。FlowDroid是基于流分析技术的隐私泄露分析工具，它对Android应用程序的生命周期做了完整建模，并构建了一个虚拟的dummyMainMethod方法来模拟生命周期，在对应用内的Sources点及Sinks点做了标记后，使用过程间数据流分析技术寻找这些点间的传播路径。

本模型基于FlowDroid所提供的分析接口以及3.1归类的Sources点与Sinks点进行Android应用中的隐私泄露分析。

3.4.3 隐私泄漏检测模块的运行测试效果

基于3.4.1中整理的Sources点和Sinks点，模型对3053款恶意应用进行了隐私泄露检测，并对检测结果做了分类整理。在这3053款应用中，总共发现了1218条窃取用户隐私的行为，统计结果如表9、10所示。

表9 泄露的隐私类别统计

|  |  |
| --- | --- |
| 隐私类别 | 在测试中的发现数量 |
| 用户位置信息 | 79 |
| 用户唯一标志信息 | 145 |
| 用户私人数据 | 176 |
| 用户网络访问相关信息 | 270 |
| 应用组件信息 | 409 |
| 程序运行间数据信息 | 139 |

表10 隐私数据发送方式统计

|  |  |
| --- | --- |
| 隐私类别 | 在测试中的发现数量 |
| 通过短信发送数据 | 388 |
| 通过网络请求发送数据 | 223 |
| 通过日志记录数据 | 267 |
| 通过本地文件文件记录数据 | 157 |
| 通过应用组件传递数据 | 183 |