今天的智能手机是私人和隐私数据的普遍存在的来源。同时，智能手机用户受到意外泄露重要数据的不小心编程的应用程序的困扰，并且通过恶意的方式来有意识地使用这些数据有意识地打印出来。虽然现有的静态污染分析方法具有提前检测这种数据泄漏的潜力，但Android的所有方法都使用许多粗粒度近似值，可以产生大量漏失和虚假警报。在这项工作中，我们提出了FLOWDROID，这是Android应用程序的一种新颖和高精度的静态污染分析。 Android的生命周期的精确模型允许分析适当地处理由Android框架调用的回调，而上下文，流程，字段和对象敏感性允许分析减少虚假警报的数量。新颖的按需算法帮助FLOWDROID同时保持高效率和精确度。推荐DROIDBENCH，无与伦比地评估Android应用程序特有的污染分析工具的有效性和准确性。通过使用SecuriBench Micro，DROIDBENCH和一系列知名Android测试应用程序的一组实验，FLOWDROID显示了非常高的数据泄漏，同时保持误报率较低。在DROIDBENCH上，FLOWDROID实现了93％的回收率和86％的精度，远远优于商业工具IBM AppScan Source和Fortify SCA。 FLOWDROID成功在Google Play的500个应用程序的一小部分中泄露出来，并且VirusShare项目中有大约1,000个恶意软件应用程序。

1. 引言

根据最近的一项研究[9]，Android手机市场的市场份额不断增长，现在已经达到了81％。随着Android手机无所不在，它们成为一个值得关注的论坛，其中包括敏感的Android应用程序中的一个主要威胁是隐私违规，泄露敏感信息，如位置信息，联系人数据，图片， SMS消息等到攻击者。但是即使是不恶意的程序，也被认为是程序化的应用程序，因为它们是广告库[16]。许多开发人员包括这样的库，为他们的努力获得一些报酬，但很少有人充分了解他们的隐私影响，也不能完全控制这些库处理的数据。公共库提供用于识别用于定向广告的人的私人信息，例如唯一标识符（例如，IMEI，MAC地址等），国家或位置信息。污染分析通过分析应用程序并向人类分析人员或自动化的恶意软件检测工具提供潜在的恶意数据流来解决此问题，然后可以确定泄漏是否确实构成违反政策的行为。这些方法通过应用程序通过从预定义源（例如API方法返回位置信息）开始，然后跟随数据流，直到达到给定的接收器（例如将信息写入插座）的方式来跟踪敏感的“污染”信息），给出哪些数据可能在哪里泄露的精确信息。分析可以动态和静态地检查应用程序。然而，动态程序分析需要许多测试运行才能达到适当的代码覆盖率。此外，当分析的应用程序执行时，当前的恶意软件可以识别动态监视器，导致应用程序在这些情况下构成良性程序。虽然静态代码分析不会共享这些问题，但是它们存在不精确的风险，因为它们需要从程序输入中抽象出来并且对运行时对象进行近似。运行时执行的精确建模对于Android应用程序来说特别具有挑战性，因为这些应用程序不是独立应用程序，而是Android框架中的插件。应用程序由具有不同生命周期的不同组件组成。在应用程序的执行过程中，框架会在应用程序中调用不同的回调，通知系统事件，这可以启动/暂停/恢复/关闭应用程序等。为了能够有效地预测应用程序的控制流，静态分析不仅要对该生命周期进行建模，而且还必须集成进一步回调系统事件处理（例如，对于诸如GPS的电话传感器），UI交互等。正如我们在这项工作中所展示的，识别回调是微不足道的，需要专门的算法。另一个挑战是敏感信息来源，例如用户界面中的密码字段。仅基于程序代码，无法检测到各自返回其内容的API调用（因为getbyid是无法知道是不是敏感信息块的）。相反，他们的检测需要存储在清单和布局XML文件中的辅助信息的模型。最后但并非最不重要的是，像任何用Java编写的应用程序，Android应用程序也包含别名和虚拟调度结构。典型的静态分析用于Javahandlethese问题通过一定程度的上下文和对象敏感性。 Android的框架性质使得这个问题比平常更难，因为我们发现它暴露了非常深的别名关系。过去的数据流分析方法针对Android [14,15,24,40]手段，在不满意的情况下，使用粗粒度过低以及低于近似值。通常由于缺乏忠实的生命周期模型造成的不足之处可能导致这些分析错失重要的数据流。实际上，尽管如此，这些工具的过度近似可能导致许多虚假警告，很容易使安全分析师压倒一切，直到他们完全停止使用分析工具。在这项工作中，我们提出了FLOWDROID，这是一款专门针对Android平台定制的新颖的静态污染分析系统，并且基于新颖的按需算法，可以保持可接受的性能而产生高精度。 FLOWDROID分析应用程序的字节码和配置文件，以发现潜在的隐私泄露，无论是由粗心造成还是由恶意创建。与之前的分析相反，FLOWDROID是第一个静态污染分析系统，它完全具有上下文，流水，现场和对象敏感性，同时精确地建模了整个Android生命周期，包括正确处理回调和用户定义的UIwidgets，并设计了最优化的精确度和召回率，即目标最大限度地减少错漏和错误警告的次数。为了在保持可接受的性能的同时获得深刻的上下文和对象敏感性，FLOWDROID使用一种新颖的按需别名分析。分析算法灵感来自仙女座[37]，但在精准度方面改进了仙女座。我们在2013年夏季开放了FLOWDROID。该工具已经被几个研究小组收录，我们与防病毒工具的领先制造商联系，他们计划在分析后端使用FLOWDROID。对于我们和其他人来说，能够测量这一重要研究领域的科学进步，研究人员需要进行Android色彩分析工具的比较研究。不幸的是，到目前为止，还没有任何可以进行系统研究的基准。作为这项工作的另一个贡献，我们因此提供了DROIDBENCH，一种新型的开源微型基准测试，用于比较Android平台的效能。我们已经使DROIDBENCH在2013年春季上线，并了解了已经使用它的几个研究小组来测量和提高Android分析工具的有效性[19]。第一组外部研究人员已经同意为套件提供进一步的微量基准[35]。 FLOWDROID可用于保护内部开发的Android应用程序，并协助分解Android恶意软件。这两种用例都不是一个完美的，而是一个合理的低误报率和假阴性。使用SecuriBench Micro，DROIDBENCH和一些包含数据泄露的知名应用程序进行的一系列实验表明，FLOWDROID在保持误报率低的同时，显示出非常高的数据泄漏。在DROIDBENCH 1.0上，FLOWDROID实现了93％的召回率和86％的精度，远远超过商业工具AppScan Source [2]和Fortify SCA [3]。实际应用程序的进一步实验证明FLOWDROID在实践中的实用性。

总而言之，这项工作提供以下原始贡献：

•FLOWDROID，第一个完整的上下文，字段，对象和流畅的污染分析，其中考虑了Android应用程序的生命周期和UIwidgets，以及特征，特别是按需别名分析的精确变体; •FLOWDROID的完全开放源代码实施

•DROIDBENCH，一款用于Android流量分析的小型，开放和全面的微型基准测试套件，•与商业工具AppScan Source和Fortify SCA相比，FLOWDROID的一系列实验提供了卓越的精确度和回忆性，以及•一组实验应用FLOWDROID到Google Play的500多个应用程序，以及来自VirusShare项目的大约1000个恶意软件应用程序[1]。我们将在线完整实现作为开源项目，以及所有基准和脚本来重现我们的实验结果：

http://sseblog.ec-spride.de/tools/flowdroid/ Spacelimitationsprecludeusfromincluded需要完全重现我们的方法。因此，我们发布了随附的技术报告[13]，其中规定了FLOWDROID的转移功能，并提供了有关实施的更多细节。本文继续如下。第2节给出了一个激励的例子和必要的背景和安全性。第3节解释了FLOWDROID如何建模Android生命周期，而第4节则提供了有关实际污点分析的重要细节。在第5节中，本文讨论实现细节和限制，第6节评估FLOWDROID。第7节讨论相关工作，第8节总结。

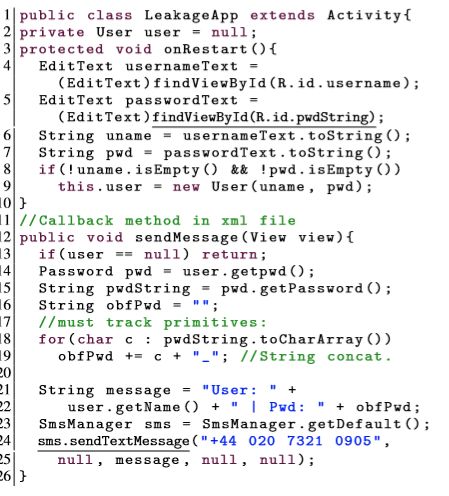
1. 背景和示例

我们首先给出一个激励的例子，然后解释这个工作所假设的攻击者模型。清单1中的示例（从真实世界的恶意软件应用程序[42]中抽象出来）实现一个活动，在Android中代表一个屏幕用户界面。每当框架重新启动应用程序时，应用程序都会从​​文本字段读取密码（第5行）。当用户点击活动按钮时，密码通过短信发送（第24行）。这构成了从密码字段（源）到SMS API（接收器）的污染数据流。在此示例中，sendMessage（）与应用程序的UI中的按钮相关联，当用户单击按钮时触发。在Android中，听众可以直接在代码中或在layoutXML文件中定义，因为仅仅分析源代码是不够的 - 分析还必须处理元数据文件以正确关联所有的回调方法。在此代码中，只有在sendMessage（）执行之前调用onRestart（）（初始化用户变量）时才会发生泄漏。为了避免假阴性，污点分析必须正确地对应用程序生命周期建模，认识到应用程序重新启动后，用户确实可以按下该按钮。为了避免误报，对此示例的分析必须敏感：用户对象包含用户名和密码的两个字段，但只有后者应该被认为是私有值。对象敏感性是必不可少的。源于不同分配站点但达到相同代码位置的对象。在我们的实验中，我们发现一些需要深层物体敏感性的情况能够自动消除假阳性。这是由于Android框架的相对较深的调用和分配链。

字符串连接（line19）等操作需要一个定义数据是否以及如何通过这些操作的模型。处理像正常方法调用和分析库方法（如应用程序代码）这样​​的操作可能是不精确的（因为它忽略了操作的语义），而且正如我们发现的，在实践中经常是昂贵的。

攻击者模型

FLOWDROID可用于一般检测数据流，无论是由粗心或恶意造成的。对于恶意案例，我们假设以下攻击者模型。攻击者可能会提供任意恶意Dalvik字节码的应用程序。通常，攻击者的目标是通过用户授予的危险广泛的权限来泄露私人数据[4]。 FLOWDROID对安装环境和应用程序输入做出了良好的假设，这意味着攻击者也可以自由地篡改这些。然而，FLOWDROID确实认为攻击者无法规避Android平台的安全措施或利用侧边渠道。此外，我们假设攻击者不使用隐式流[20]伪装数据泄漏。鉴于当前可用的恶意软件，这是一个非常合理的假设。



3.生命周期的精确建模

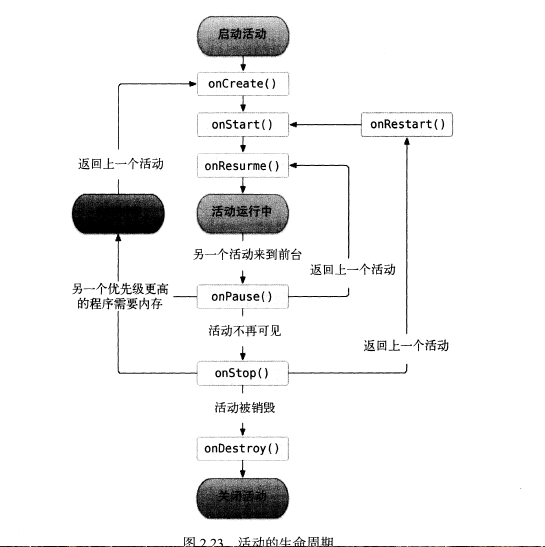
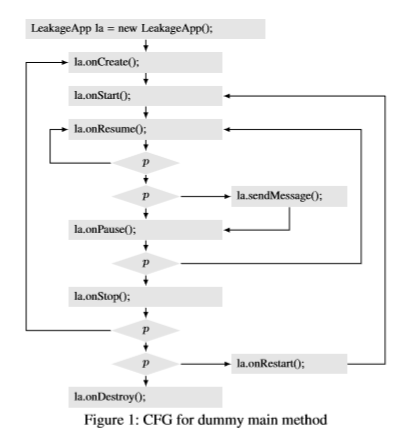
在下文中，我们将介绍FLOWDROID对生命周期的精确建模，包括入口点和异步执行的组件和回调。

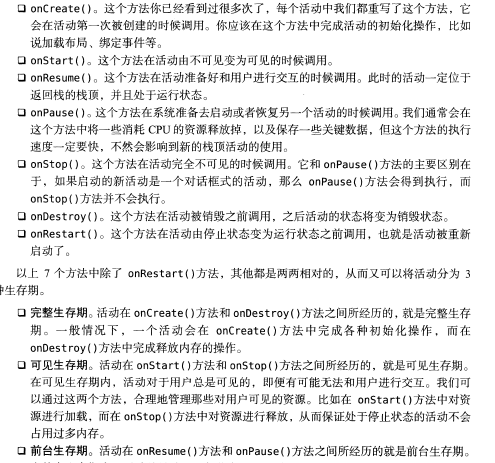
**多个入口点**

与Java程序不同，Android应用程序没有主要的方法。应用程序包含许多入口点，即由Android框架隐式调用的方法。 Android操作系统为应用程序中的所有组件定义完整的生命周期。应用程序开发人员可以定义四种不同类型的组件：活动是单一关注的用户操作，服务执行后台任务，内容提供商定义数据库类存储，广播接收方监听全局事件。所有这些组件都是通过从预定义的操作系统类派生自定义类来实现的，在AndroidManifest.xml文件中注册并覆盖生命周期方法。 Android框架根据环境需求调用这些方法来启动或停止组件，或者暂停或恢复组件。例如，它可以因为内存耗尽而停止应用程序，然后在用户返回时重新启动应用程序[17]。结果，构建调用图时，Android分析不能简单地从检查预定义的“主”方法开始。相反，Android生命周期中的所有可能的转换必须精确地建模。为了解决这个问题，FLOWDROID构造了一个模仿生命周期的自定义模糊方法。下面我们来解释一下如何构造这个方法。

**异步执行组件**

应用程序可以包含多个组件，例如三个活动和一个服务。虽然活动顺序运行，但不能预先确定。例如，一个活动可能是最初对用户可见的主要活动，然后根据用户输入启动其中一个活动。服务作为并行后台任务运行。 FLOWDROID通过假设应用程序内的所有组件（活动，服务等）可以以任意顺序（包括重复）顺序运行来建模此执行。一些静态分析是路径敏感的，即分别考虑每个可能的程序路径。在这种情况下，考虑所有可能的排序将是非常昂贵的。 FLOWDROID基于对IFDS [32]的分析，这是一个不是路径敏感的分析框架，而是立即在多个控制点之间加入分析结果。 FLOWDROID生成和有效地分析虚拟主方法，其中单个组件生命周期和回调的每个顺序都是可能的;它不需要遍历所有可能的路径。





**回调**

Android操作系统允许应用程序注册各种类型信息的回调，例如位置更新或UI交互。 FLOWDROID在其虚拟主方法中对这些回调进行建模，例如用于识别应用程序将框架传递给回调的位置数据作为参数进行存储的情况，然后在活动停止后发送此数据。调用回调的顺序通常不能预测，这就是为什么FLOWDROID假定所有回调都可以按任何可能的顺序调用。但是，回调扫描仅在父组件（例如活动）正在运行时发生。为了精度，FLOWDROID因此将组件（活动，服务等）与他们注册的回调相关联。例如，当按下按钮时，可以注册被调用的回调。然后，必须在此活动的onResume（）和onPause（）事件之间分析相应的回调处理程序。

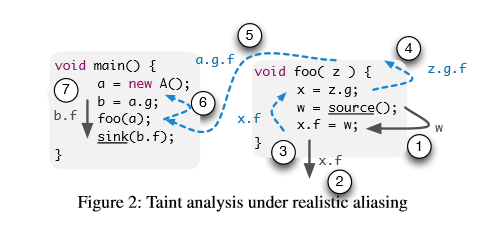
在Android平台上注册回调处理程序有两种不同的方法。首先，回调可以在XML文档中以声明方式定义。或者，也可以使用公知的特定系统方法调用注册。 FLOWDROID支持两种方式。此外，对于恶意软件，攻击者可能会通过覆盖Android基础架构的方法来注册未记录的回调，其中一些甚至可以通过本地代码调用。 FLOWDROID识别这样的覆盖方法，处理它们与普通回调处理程序类似，例如按钮点击。

为了寻找在应用程序代码中注册的回调，FLOWDROID首先从相应组件类中实现的生命周期方法（onCreate（），onStop（）等）开始，计算每个组件的调用图，然后，这种调用图用于对那些使用公知的回调接口作为回调接口的安卓系统方法扫描声明，之后，调用图被逐渐扩展以包括这些新发现的回调，并且扫描再次运行，因为回调处理程序可以自由注册新的回调，这可能需要FLOWDROID重新扩展调用图并重新分析，直到固定点达到。虽然这种方法比只是扫描实现回调接口的类更昂贵，但它可以在组件和回调之间提供更精确的映射。这不仅减少了误报，而且我们还发现它大大减少了以下污点分析的运行时间。一旦构建了虚拟主方法，FLOWDROID使用该方法计算最终调用图，作为应用程序的入口点。对于布局XML文件中定义的回调，相应的XML文件将使用相应的布局控件映射到一个或多个应用程序组件（什么叫布局文件中定义的回调呢？想象我们在基于各种系统生命周期的回调方法开始分析，然后扩展到其他的自定义回调方法，但是这其中并不包括对xml控件上的回调方法吧，他们应该是放在了其他专门的函数里，所以这种就需要去控件里寻找有那些控件做了回调方法）例如，按钮单击处理程序仅适用于托管相应按钮的活动。 FLOWDROID分析seach活动，以查看其注册的XML文件中的哪些标识符。然后，此信息用于创建映射。

例

请注意，为了获得最大精度，FLOWDROID为分析的每个应用程序生成一个新的虚拟主方法。 每个主要方法只涉及生命周期的一部分，根据应用程序的XML配置文件，实际上可以在运行时发生。 禁用的活动被自动过滤，并且回调方法仅在它们实际属于的组件的上下文中被调用。 例如，按钮点击处理程序仅在其各自的活动的上下文中进行分析。在图1中，我们显示了上一个示例的虚拟主方法的控制流图。 图表模拟了通过sendMessage回调增加的通用活动生命周期。 在这个图中，p表示一个不透明的谓词，我们知道FLOWDROID将无法静态地进行评估。结果，分析将自动考虑两个分支涉及p的条件。

1. 精准的流敏感分析



分析中的一个主要困难是如何有效地实现高对象灵敏度来解决混叠。 图2（从现实世界的情况中抽象）显示了FLOWDROID如何组合一个前进的分析和一个按需的后向别名分析来推断出b.f在汇点处被污染。 在步骤1中，污染的变量w被传播前进，污染堆对象x.f.Step 2继续跟踪vwx的重要步骤重要的步骤是3：每当堆对象受到污染时，向后的分析向上搜索相应对象的别名 （在这种情况下为xf）。 在7处，找到别名b.f，然后作为正常污点向前传播。

FLOWDROID模拟IFDS [32]框架内的程序间分布子集问题的污染分析问题。 第4.1节解释了分析使用的传递函数。 大多数功能都比较标准。 然而，有一个重要的情况，其中FLOWDROID的分析与标准污染分析算法不同，即在污染值被分配给堆的语句，即对于字段或数组。 这种情况将导致调用向后的别名分析，（从图里面可以很清晰的看到，观测到w作为污点后，接着走，观察w是否传播给了其他的变量，以及是否涉及到了sink点，注意如果传播给的对象是堆的话，也就是一个对象的话，就需要进行向后的别名分析，直到找到堆对应的别名，因为堆的话可能会赋值给其他的变量的）详细的说明将在4.2节中进行说明。 由于空间限制，我们将流程功能的描述保持在非正式的层面。为了允许其他人重现我们的方法，随附的技术报告包含完整的正式化

4.1污点分析

前向和后向分析均传播访问路径。访问路径的形式为x.f.g，其中x是局部变量或参数，f和g是字段。访问路径可以具有不同的长度，可由用户定制的最大长度（5bydefault）。长度为0的访问路径是简单的局部变量或参数，例如x。在FLOWDROID中，访问路径隐式描述可通过该路径访问的所有对象的集合，例如，x.f包括x.f.g，x.f.h，x.f.g.h等等。如果右侧的任何操作数被污染，则作业的传递函数将显示左侧。通过污染整个数组来保护数组元素的赋值。向变量x分配“新”表达式将擦除以root为基础的访问路径建模的所有色彩。方法调用通过使用形式参数替换实际来转换访问被调用者上下文的访问路径;逆转换发生在方法返回，包括返回值（如果存在）。与通常的基于IFDS的分析一样，FLOWDROID还包括一个呼叫转移流程功能（绕过呼叫者侧面的每个方法调用）。该功能传播与呼叫无关的色调，在源处生成新的色调，在接收器处报告阴影并传播本机调用的色调。第5节进一步提供了后者

4.2按需别名分析

每当一个污染的值被分配给诸如字段或数组的堆位置时，FLOWDROID向后搜索目标变量的别名，然后也会污染它们。 在清单2中，现在考虑第一次调用taintIt（第3行），这样会导致形式参数的变化。在第10行中，这将导致访问路径xf由于分配xf = in而被污染。在这种情况下 通常在堆的所有分配上），FLOWDROID将启动对xf的别名的反向搜索，找出.finline9.在这一点上，对于out.f从这个语句开始一个新的前向污染传播，这将最终发现泄漏 第11行。反向分析也将继续向后搜索，发现主要的别名pf，然后产生一个前进的分析，导致第4行的第二个流行度报告。