Android安全：恶意软件的渗透以及防御问题的研究

计121 10123522 胡伟强

摘要：智能手机已经普及。这不仅是因为这些常规服务如语音呼叫，发短信，和多媒体服务。而且是因为如办公应用，上网，手机游戏，以及基于GPS的车辆导航服务的新兴应用。Android设备已经获得了巨大的市场份额，这是由于Android系统的开放式架构和应用程序编程接口在开发者社区的日益普及。在2010到2014年之间由于和Android相关的收益大幅增加吸引了大量的恶意软件开发者进行恶意软件开发，造成Android恶意软件的大幅增加。专业的学术研究人员和商业反恶意软件公司已经意识到，传统的基于签名和静态分析方法的应用程序是脆弱的。特别是，盛行的隐形技术，如加密，代码转换和环境感知等方式，能够产生已知恶意软件的变种。这已经导致了以动态分析为基础的方法的使用。用单一的方法对抗先进的技术可能是无效的，多元互补的方法可以在检测串联有效恶意软件中使用。现有的讨论已经广泛覆盖了智能手机操作系统的安全性。然而，我们认为，Android的安全性，尤其是随着恶意软件的增长，自动分析技术的研究，和现有的检测方法，需要一个广泛的覆盖。在本次研究中，我们讨论了Android安全保护机制，以及现有的安全保护机制所面临的威胁，和在2010年到2014年间隐形的恶意软件的增长时间表，以及其所应用的恶意技术和检测方法。本文综述了现存的恶意软件检测技术的优缺点，并为研究人员和从业人员提供了一个面向下一代恶意软件检测技术研究的平台。

关键词：Android恶意软件，静态分析，动态分析，行为分析，模糊化，隐形恶意软件。

1. 导言

Android智能手机操作系统已经占据了整个市场份额的75％以上，使得其竞争对手iOS，Windows移动操作系统和黑莓手机远远落后。甚至虽然智能手机在过去十年就已经被使用，然而直到2008年，iOS和Android智能手机操作系统的诞生才对全世界的用户和开发者产生了巨大的吸引力。由于广泛的连接选项技术的发展，如GSM，CDMA，无线网络，GPS，蓝牙和NFC，智能手机已经变得无处不在。 Gartner 2013年的智能手机销售报告显示Android设备销量比上年同期增长42.3％，。整体市场份额由66％上升至78％，用户数大幅上涨12％。于此相对应Android的竞争对手iOS的销量却下降了4％，从19％降至15％。无处不在的互联网连接和个人的隐私信息，如联系人，邮件，社交网络访问，浏览器历史记录和信用卡记录吸引了恶意软件的开发者对移动设备的关注

Android尤其如此。 Android的恶意软件，如保险费率短信木马，间谍软件，僵尸网络，攻击性广告软件和特权升级攻击软件，正利用Google Play 和其他第三方市场场分发。Android的普及，鼓励开发者提供了很多俗称app的应用程序。官方Android应用市场Google Play，承载第三方应用程序会象征性收费。Google Play主机承载了每日超过百万下载量的应用程序。和苹果AppStore不同的是Google Play不会手动验证上传的应用程序。相反，官方市场依赖与“蹦床”，一个为了应对恶意程序威胁的动态仿真环境。虽然“蹦床”可以防止恶意软件的威胁，但是它不能不分析上传app中的漏洞。恶意软件作者利用这些应用程序的漏洞泄露用户隐私信息进行谋利，这在无意中伤害了应用商店和开发商的声誉。此外，Android的开源理念允许第三方应用程序市场中的app安装，这其中混杂了数十个地区和国家的应用 -商店。然而充分的保证应用程序质量的方法对第三方应用商店的存储有是一个挑战。

Android安全解决方案供应商报告一个惊人的研究报告。在2010年，对100个软件样本的研究发现只有三个恶意软件。而这个数量却在2013年第四季度成倍增加。恶意软件作者使用隐藏技术，动态执行，代码混淆方法，重新包装和加密技术，以绕过现有的Android平台和商业公司提供的保护机制。现有的恶意软件通过使用上述的技术进行传播打败了传统的基于签名的的技术。这些适应了智能手机平台并提供及时响应的新技术对Android平台是迫切需要的。对于未知恶意软件的频繁签名更新的主动检测技术对Android系统是迫切需要的。

恶意软件开发者利用平台的漏洞获得智能手机的控制权，并盗取敏感的用户信息，以希望利用电话服务或创建僵尸网络获利。因此，了解他们的业务活动，工作模式和保护移动设备的主动监测设备的使用模式，是重要的。

指数增加的恶意应用程序，迫使反恶意软件行业开发出稳健有效的适合现存的制约因素之内的检测方法。

现有的反恶意软件解决方案提供商采用基于签名的检测，这是由于其执行效率高和操作简单。基于签名的方法可以很容易地被恶意软件使用代码产生一个新的签名而规避，迫使反恶意软件客户定期更新签名数据库。由于有限的处理能力和电池受限的可用性，基于云的分析和检测解决方案已经出现。人工分析和提取恶意软件签名需要足够的时间和专业知识。它也可以利用生成签名族的变种产生假阴性（FN）。由于恶意软件变种成指数增加，有必要采用自动生成签名的方法降低误报率。

“Off-device”恶意软件分析方法需要了解恶意软件的功能。为了提取恶意软件签名需要对样品进行手动分析。然而，考虑到恶意软件的迅速崛起，迫切需要有最低人为干预的分析方法方法。 自动分析有助于对检测看不见的恶意软件分析生成及时响应。静态分析可以快速准确地识别恶意软件的模式。然而，它对代码转换，本机代码和Java反射是无效的。因此，动态分析的方法，虽然耗时，但是它是通过让恶意软件在沙箱环境执行以提取隐匿恶意软件的恶意行的替代。

学术界和工业界的研究人员提出了分析和检测Android恶意软件威胁的解决方案和框架。其中一些甚至是开源的。这些解决方案可以使用以下三种被表征参数：

1. 所提出的方案的目标是评估，分析或恶意软件检测。应用程序安全评估解决方案，确定了被对手利用危害用户和设备安全的安全漏洞。分析解决方案检查未知的应用程序中的恶意行为，而检测解决方案的目的是防止其在设备上安装。

2）方法上为实现上述目标可以是静态或动态分析为基础的恶意软件检测方法。控制流和数据流分析是正式的静态分析方法的例子。在动态分析方法中，

为了监测其活动，并确定异常的行为，应用程序的执行/模拟都是在沙箱环境中进行的，若用静态分析方法是非常困难的。

1. 部署上面讨论的解决方案。

现有的智能手机安全调查研究都只考虑到流行的手机操作系统平台。然而，本文则侧重于最流行的移动设备操作系统Android平台。La Polla等，在2004-2011年期间调查了智能手机的安全威胁及其解决方案，但对Android系统的覆盖非常有限。

Suarez-Tangil La Polla等人的工作集中在智能传感器基于特征的误用攻击，如硬件，通信，传感器和系统。作者给出了对于Android系统的滥用对安全性影响的见解。作者基于恶意软件的攻击目标，分发方法，感染和特权采集进行了分类。相反，本文根据2010 - 2014年之间商业反恶意软件行业术语对其进行了分类并提供了恶意软件感染率和威胁的准确描述。

2011年，威廉·恩克研究了Android安全机制，通过权限特别保护和跨应用程序通信的安全性问题。此外，笔者讨论了其他的第三方Android平台硬化解决方案，已经其优点和局限性。此外，此次研究还检查了app的安全性分析建议，并提出了未来的发展方向，以提高Android平台的安全性。

本文旨在通过扩展Android安全问题的报道和恶意软件在2010-14之间的增长来补充前者的评论。本文讨论代码转换方法和强度，以及恶意软件分析和检测方法的局限性。尤其是，

本文全面涵盖了用于恶意软件作者通过生成已知恶意软件的变种逃避检测的隐身技术。最后，我们提出了一个混合了Android恶意软件分析和检测的框架，以深入了解我们未来的研究方向。本次文的结构如下。

第二节讨论了用来削弱攻击Android app体系结构和安全执行机制。第三节覆盖了第二节讨论的现有的执法行动所面临的Android安全问题。第三节覆盖了Android新版本中为解决现存安全问题的安全增强功能。第四节呈现了2010 - 2013年Android的恶意软件根据他们的分类和功能的时间线。第五节介绍了各种渗透并受雇于先进的Android隐身技术的恶意软件。

第六节和第七节根据它们的部署解决方案的检测方法进行归类评估，分析。第八节根据在第六节提及的功能以及提出的学术界和反恶意软件行业最先进的工具的状态进行分类。 本部分讨论众所周知的分析技术的优点，缺点，工具结果见表一。流行的基于web的分析沙箱见表Ⅱ。最后，第九节

总结了本文提出的一种混合型恶意软件分析检测框架作为未来的建议和研究方向。

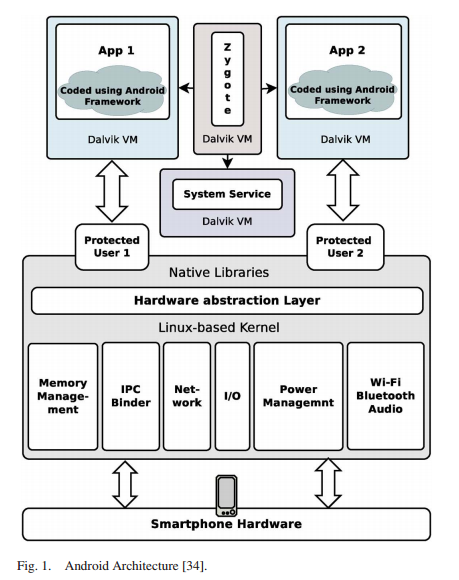
1. Android app与安全架构

Android是由Android开源开发项目（AOSP）开发，由谷歌持有其所有权，由开放手机联盟（OHA）所推广。开放手机联盟由原始的设备制造商（OEM），芯片制造商，运营商和应用程序开发人员所组成。Android应用程序是用Java编写的，但是本机代码和共享库的开发由C / C ++完成。典型Android架构示意图如图1。底层的Linux内核是为有限的资源的嵌入式环境而特别定制的。安卓

是基于Linux内核之上，这是由于其强大的驱动程序，高效的内存和进程管理，网络化为核心的服务支持。目前，Android支持两种指令集体系结构：1）ARM，在智能手机，平板电脑非常普遍; 2）X86，其中在移动互联网设备（MID）盛行。在Linux内核的上面，本地库由C / C ++开发，其支持高性能且可被第三方重复使用。Android的用户应用程序，用Java语言编写，运行时被转换为Dalvik字节码，Dalvik虚拟机（DVM），如图1所示。它对资源受限的移动操作系统平台进行了优化。一旦操作系统启动完成，一个被称为合子初始化Dalvik虚拟机的进程就由预加载的核心库加载。合子然后通过一个socket等待加载新的派生进程。合子进程加快应用程序加载库的实例，以与新加载的用户程序共享。最后，应用程序框架为Java库开发者提供了一个均匀的简练的视图。Android利用服务与许可为主的模式提供保护敏感信息的功能，如电话，GPS，网络，电源管理，广播和媒体系统等。

1. App 架构

Android应用程序被打包成一个叫做APK的.apk文件，一个由若干文件和文件夹组成的zip压缩包，如图2所示。特别是，AndroidManifest.xml中存储的了很多元数据，如包名，所需的权限，一个或多个actuvity，如intent，service，广播接收器和内容提供商，最小和最大的版本，支持库链接等。



Res文件夹存储图标，图片，字符串串/尺寸/颜色常量，UI布局，菜单，动画等都被编译成二进制文件。Assets 目录 包含了不被编译的资源.可执行文件classes.dex存储的要在Dalvik虚拟机执行的Dalvik字节码。META-INF存储用来核实第三方身份的应用程序开发人员证书的签名。正如前面提到的，Android app用Java开发。开发过程如图3所示。编译Java代码生成多个.class文件，这是由Java源代码中定义的类的数量所决定的。运用

DX tool，.class文件被合并成一个单一的Dalvik可执行的（.dex）文件。为了加速执行，.dex文件存储了要在DVM执行的Dalvik的字节码。

B. App 组件

一个Android应用程序是由一个或多个组件组成：

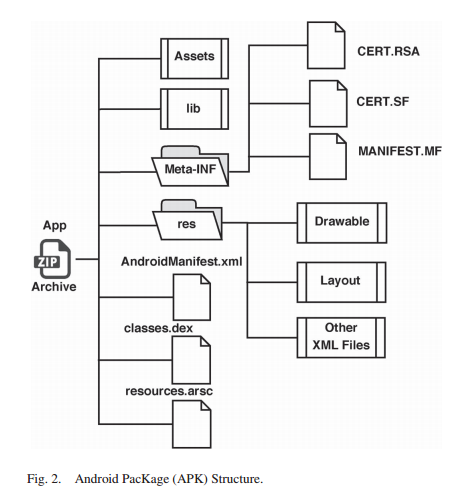
•Activity：它是一个应用程序的用户界面组件。根据开发者的需求任何数量的activity都可以在manifest文件中配置。除了一些预先定义的任务，一个活动也可返回结果给它的调用者。Activity是由将要在C中解释的Intent所启动的。

•Service：service组件执行后台任务，没有任何用户界面。例如，播放来自网络的音频或下载数据。Service是由将要在C中解释的Intent所启动的。

•广播接收器：该组件监听Android系统生成的事件。例如，BOOT\_COMPLETED，SMS\_RECEIVED等都是系统事件。其他应用程序可以广播自己的应用程序定义的事件，可以通过其他应用程序的service组件进行处理。

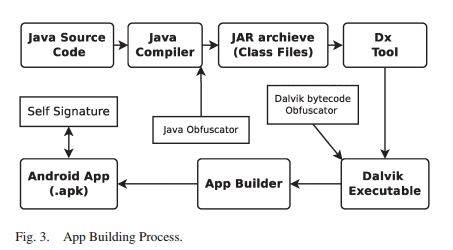
•内容提供者：内容提供者也被称为数据存储，提供了一个供相同和不同应用程序之间进行数据访问的一致的接口。从外部看，内容提供者中显示数据关系。然而，它可以具有一个完全不同的存储实现。数据存储是通过应用程序定义的统一资源标识符（URI）访问的。

其他应用程序可以访问其他应用程序明确其导出的组件。清单1讨论组件的声明作为使用例子定义的AndroidManifest.xml。声明的组件可以被独立调用或执行，应用程序组件的开发和传播是异步的。Android应用程序有多个入口点，这取决于定义的组件的应用程序的数量。



C.组件间的通信

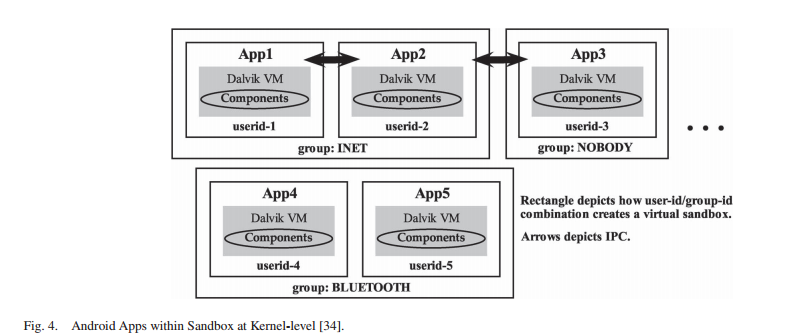
Android安全与保护相结合的应用程序和数据系统级组件间通信（ICC）的。ICC定义使用的保证核心安全是基于Linux的框架的。一个应用程序运行了一个独特的用户ID，阻碍编程发布。Android的中间件介导ICC在应用程序和组件之间。通过分配接入许可标签限制访问一个组件。当一个组件发起ICC，参考监视器着眼于分配给它的容器应用程序的权限标签。如果目标组件访问权限标签在该集合中，它允许ICC被启动。如果标签不属于集合，ICC初始化将被拒绝，即使组件是应用程序的一部分。开发人员通过Manifest文件分配一个应用程序的权限。开发人员定义该应用的安全策略，而分配许可中的组件应用程序指定访问策略来保护其资源。应用组件彼此在一个高层次抽象的相互作用通信（IPC）机制Intent进行进程间通信的，这是被IPC驱动程序处理的。应用程序是通过Intent进行调用活动和服务，并发送广播的。系统事件也是通过Intent广播的。Intent可以使用包含接收器组成的由类/包名称字段组成的明确地址。取决于存在动作类和数据字段，系统将隐含意图到一个或多个匹配接收器的组件。每个组件将自己注册为接收使用了一个或多个Intent过滤器的Intent。它还规定，如果一种行为，种类和/或数据可以由意图被接受。如图所示的清单1中，当它接收到该系统服务组件仅调用意图与行动等于BOOT\_COMPLETED在行号29。



D.应用程序沙箱

Android已经被设计为安全的移动操作系统，以保护用户数据，开发者的应用程序，设备和网络为动机。但是，安全依赖于显影剂意愿和能力，坚持最好的发展实践。此外，用户必须了解应用程序可对数据和设备的安全性的影响。反恶意软件解决方案没有足够的权限来执行攻击性的恶意软件检查由强制操作系统的安全模式。例如，反恶意软件应用程序都受限制的扫描或监视能力或使用文件系统中的设备。本节介绍Android的安全功能。Android的内核实现了Linux的自由访问控制（DAC）。每个应用程序处理被保护与分配唯一的ID（UID）在一个隔离的沙盒。沙盒

抑制干扰其他应用程序或其他的应用程序的系统服务。Android的保护网络访问通过实现一个功能偏执网络安全，一个功能控制无线网络连接，蓝牙上网。如果一个应用程序具有访问网络资源的权限（如蓝牙），该应用过程被分配给相应的网络接入ID。因此，除了UID，一个过程可以被分配一个或多个组ID（GID的）。 Android应用程序沙盒说明见图4。



一个应用程序必须包含与开发商键（参见图3）签署的PKI证书。应用程序签名是信任的点谷歌和第三方开发者，以确保应用程序的完整性和开发商的声誉。应用程序签名的程序放置一个应用程序到一个隔离的沙盒分配一个唯一的UID。如果一个应用程序A的证书与已安装的应用程序b匹配在设备上，Android的分配相同的UID（即沙箱），以应用程序A和B，允许他们分享他们的私人文件和Manifest文件中定义的权限。这意外的共享可能由恶意软件编写者天真开发者利用可能生成两个证书。这是最好的对于开发者保持自己的私人证书以避免其滥用。

E.框架级的权限

为了限制一个应用程序访问敏感功能，如电话，网络，通讯录/短信/ SD卡和GPS定位等，安卓在应用框架上提供基于权限的安全模型。开发者必须用权限标签在AndroidManifest.xml中声明所需的权限，如前讨论。安卓控制个人应用程序，以减轻系统应用程序或第三方开发者应用程序内的不良影响。这些限制在安装过程中被强制执行。安卓权限分为以下四个保护等级：

1. 正常：这些权限在用户，系统应用程序或设备上具有较低的风险。正常权限在安装时默认分配。
2. 危险：这些权限属于高风险的，由于他们有能力访问私人数据，以及设备的重要传感器。用户必须在安装时接受潜在的危险权限。
3. 签名：仅当请求应用程序与同一开发者证书签署时才授予这些权限声明的权限的应用程序。他们在安装时是被自动授予的。签名权限对于系统应用程序也是有限的。
4. signatureOrSystem：如果申请程序与相同的证书签署安卓系统映像或使用一个应用程序这些权限将被授予，声明这个权限。他们在安装时自动获得。

安卓权限是粗粒度的。例如，INTERNET 权限没有能力限制访问特定的统一资源定位器（网址）。READ\_PHONE\_STATE权限允许一个应用程序是否设备环或正在举行。同时它也允许应用程序读取设备标识符等敏感信息。WRITE\_SETTINGS，CAMERA 权限，被广泛定义，从而违反了最小特权访问原则。访问WRITE\_CONTACTS或WRITE\_SMS并不意味着进入READ\_CONTACTS或READ\_SMS权限。因此，权限是不分层的，它们必须是单独的被开发人员定义的。在安装时，用户被迫授予所有权限或不同意应用程序安装。因此，在安装时危险的权限是无法避免的。此外，用户不能区分可能暴露的必要性和它的命令滥用为了开发。

F.安全系统分区

安卓系统分区是从内核、系统图书馆，安卓运行时，应用程序框架和应用程序。安卓系统分区是只读的为了防止未经授权的访问或修改。也有部分如应用程序缓存和SD卡是受保护的文件系统用适当的特权，以防止其被篡改对手时，设备连接到桌面上USB。

G. 安全Google Play

谷歌由于安全问题建议用户防止安装任何第三方市场的应用程序。然而，它仍然允许安装第三方市场的程序。第三方开发商的应用程序，可从官方PlayStore。谷歌的第三方开发的应用程序审查一个沙箱环境动态分析阻止任何恶意软件进入Google Play。保镖，如果不是无敌是一个合理有效的安全机制。安卓有一个验证服务的设施同时安装应用程序从其他市场的地方。Google Play如果发现恶意行为，可以远程安装。然而，该设备可用于连接到互联网的设备。

1. Android 安全问题及改进

本节对用户和设备安全问题进行了详细的讨论。此外，还涵盖了在随后更新的Android版本中的各种增强AOSP。

1. 安卓威胁

AOSP是致力于一个安全的Android智能手机操作系统，但它也容易受到社会工程学攻击。一旦该应用程序被安装，它可能会对设备安全造成不良后果。以下是已报告的或可在随后的安卓版本中应用的恶意活动列表。

•特权升级攻击被利用利用公开可用安卓内核漏洞获取设备的根访问。可利用安卓出口组件来获取危险权限。

•当用户授予恶意应用程序危险权限，恶意软件可以在没有用户的知识或同意，允许其不知不觉中访问敏感数据，这将发生隐私泄露或个人信息被盗。

•恶意应用程序可以无需用户同意通过监控语音通话、短信/彩信、银行mTANs，记录音频/视频监视用户。

•恶意应用程序可以不经过用户理解或同意，通过调用或订阅短信而赚钱。

•妥协的设备作为一个BOT和远程通过发送各种命令来控制它执行恶意活动。

•积极的广告活动可以吸引用户下载潜在有害程序（PUA），或恶意的应用程序。

•共谋攻击发生时，一套程序，签署同样的证书，安装在设备上。这些应用程序将与其他应用共享UID，即使是任何危险的请求，由一个应用程序的权限将被共享恶意串通的。总的来说，这些应用程序执行恶意的活动，但是它们的主要功能是良性的。例如，具有 READ\_SMS权限的应用程序可以读取短信，然后串通具有INTERNET 权限的伙伴向远程服务器过滤敏感信息。

•拒绝服务攻击可以发生在应用程序过度使用已经有限的CPU、内存、电池和带宽资源和限制用户执行正常功能。

1. Android 版本更新问题

以谷歌为首的Android开源项目（AOSP），进行Android源代码的升级和维护。然而，补丁，更新或升级版本的主要责任在于原始设备制造商（OEMs）或无线运营商。个别的原始设备制造商更新相应的自定义版本的操作系统。在一些国家，无线运营商为满足自己的要求定制的代工操作系统。因为这样一个更新链，所以在补丁到达最终用户之前需要几个月的时间。这种现象被称为碎片，在不同版本的安卓仍然由于分散不更新。具体而言，手机随着旧的和未修补的版本仍然容易受到已知漏洞的攻击。与桌面操作系统相比，安卓系统的更新和升级更频繁。自2008九月推出以来安卓发布了29个稳定的操作系统的版本更新和升级。在空中（OTA）更新明显变化现有的版本修改了大量的文件该平台，维护现有用户数据的完整性和应用。通过服务提供新的版本更新被称为软件包管理系统（PMS）。Xing等，进行了一个全面的Pileup漏洞研究反过来，可以利用恶意软件作者在

版本升级。一个应用程序开发的旧版本可以被用来使用危险的许可（的）介绍

版本发布。在更新过程中，安卓不验证更新应用程序中的附加权限。因此，它妥协的设备安全。在一次重大更新或升级，大量的文件被修改，以确保敏感的用户信息保持不变，在更新过程中导致复杂性。

1. 本机代码执行

Android允许通过在C / C + +使用原生开发套件库实现执行本地代码（NDK）。

尽管本地代码执行在Dalvik虚拟机，它是沙箱通过用户ID /组ID（S）组合。不过本机代码具有执行权限升级的潜力通过利用平台的漏洞，表现出相当多的恶意软件攻击在最近的过去。

1. 在最近的版本中的安全性增强

在安全问题，漏洞和/或报告恶意攻击，AOSP版本的补丁、更新、改进和升级。在这里，我们讨论了显着的安全性修正特点在随后的安卓系统版本注册成立Android 4.4：

1）安卓防栈缓冲和整数溢出操作系统版本1.5。在2.3版本中，安卓固定串

格式的漏洞，并增加了硬件基础执行（NX）支持停止在堆栈中执行代码和堆。

1. 安卓4地址空间布局随机化（ASLR）被添加到防止回libc和内存相关的攻击。
2. 信息可以前通过连接装置的电脑使用Android调试桥过滤（ADB）驱动程序。虽然开发开发作为调试工具，它允许应用程序安装，阅读，即使设备被锁定，系统分区等连接到个人电脑（电脑）。防止这种未经授权的访问，Android 4.2.2认证亚行使用RSA密钥对连接。用户响应是提示在设备屏幕上，如果亚开发访问设备。因此，如果设备被锁定，攻击者将无法获得控制。
3. 为了防止恶意软件发送收费短信的短信，Android 4.2引入了一个额外的通知功能，在用户应用程序发送短信之前以提示用户。
4. 安卓推出了一个主要能力为4.2版（17版）以允许建立多允许用户访问共享设备的多个用户如片剂。在2013七月发布的Android 4.3版本中(18)添加了限制配置文件访问的能力。这些修改被放置在介意共享移动设备如平板电脑的使用为单个用户提供私人空间移动设备。对于每个用户，一个单独的帐户，用户选定的应用程序，自定义设置，私人文件和私人用户数据分配。这种能力使多用户共享一个单一的设备。在“亩”的情况下，主要帐户是设备的所有者。使用设备设置，业主可以创造额外的“亩”。除了原始用户，其他创建“亩”用户不能创建，修改或删除设备“亩”用户。
5. Android 4.3移除了setuid() / setgid()程序，这是由于他们很容易受到伤害。
6. Android 4.3增强了SELinux提供的安全性。Android 4.4介绍SELinux执行多个根模式的过程。SELinux强制访问控制（苹果）政策代替传统的自由裁量访问控制（DAC）。在DAC，对资源的所有者决定哪些其他感兴趣的科目可以访问它，在苹果系统（不是用户）授权的访问特定资源的主题。因此，苹果即使是有根设备被损害的可能性，它仍然有防止恶意活动的潜能。因此，苹果大幅降低内核级权限的效果升级攻击。
7. 第三方安全性的增强

许多独立的Android安全性增强被提出。这些机制允许一个组织为他们的雇员创建细粒度的安全策略装置。上下文信息，如设备位置，应用程序权限和跨应用程序的通信可以监测并对已经宣布的政策进行了验证。这一范围的论文主要研究安卓系统的安全性、恶意软件的问题及国防技术，它不检查上面提到的预防技术。

1. 报告的安卓恶意软件威胁感知

图5显示了在2010-2013间一些显著的恶意软件族的时间线。其中，短信木马做了主要贡献；其中一些甚至感染了谷歌PlayStore。大量的恶意软件的应用程序利用根基于攻击如愤怒笼，越狱和root获得超级用户控制设备的权限。最近的安卓漏洞是主密钥攻击，其中有从4.2.2版本开始脆弱。每季度，反恶意软件公司报告在新的恶意软件族和现有的恶意软件指数增加变种。这些公司在近似得出了在安卓设备上的恶意软件感染率。特别地，Lookout Inc公司报告了全球恶意软件感染率的可能性百分比为2.61%。两个独立的报告研究估计实际感染率。1）作者用智能手机域名解析（DNS）感染率，其中美国报告了0.0009%感染率。ruong等人。仪表64克拉的应用估计三种不同的恶意软件数据集的感染率McAfee移动沙箱数据集的0.26%和0.28%分别。因此，目前安卓威胁感知恶意软件感染率有一个巨大的报告之间的变化商业反恶意软件和独立研究。在下面的段落中，我们讨论了安卓恶意软件分类及其特征。

1. 木马

木马伪装成良性的应用程序，但他们未经用户同意或知识做有害的活动。木马泄漏机密的用户信息，或者他们可能“钓鱼”用户和窃取的敏感信息，如密码。

直到2012第二季度，多数android恶意程序变种属于各类短信木马家族。短信木马程序是无需用户的了解和同意便发送收费短信使得用户产生财务损失。除此之外，木马还透露联系人信息、IMEI和IMSI号码来指挥

控制域。fakenetflix 将自己伪装成热门Netflix的应用程序，网络钓鱼用户输入他们的登录凭据。fakeplayer，zsone 和是Android。foney几值得注意的安卓木马给用户带来经济损失。关于增加移动银行业务的帐户，恶意软件作者有针对性的双因素移动银行认证。在捕捉目标帐户的用户名和密码，采用社会工程攻击，Zitmo和spitmo木马监视和窃取mTANs（移动交易认证号码）默默地完成交易。

1. 后门

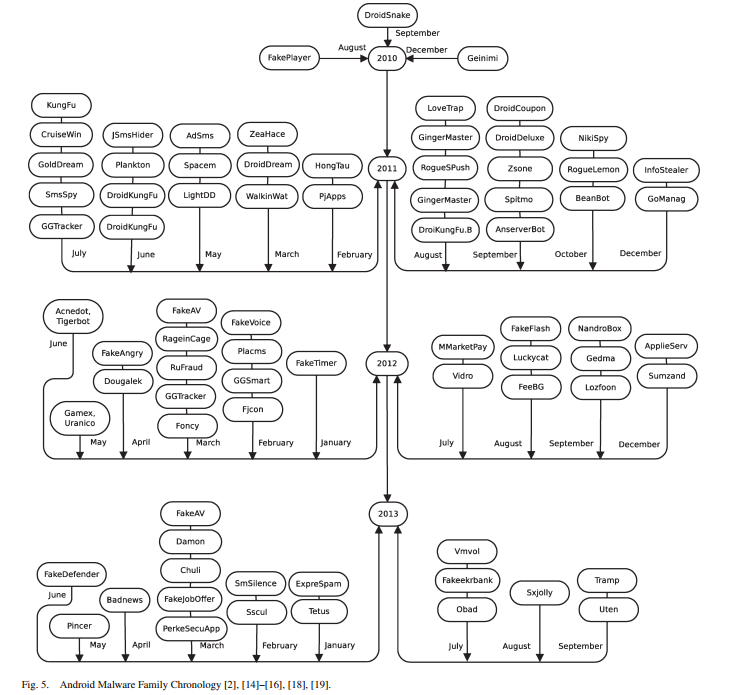
后门促进他们绕过正常的安全程序允许其他恶意软件悄悄进入系统。后门可以采用根漏洞获得超级用户权限和隐藏的反恶意软件扫描。根数利用如愤怒笼，rageagainstthecage和gingerbreak 获得完全控制设备。BaseBridge，KMin ，Obad 是已知的著名的后门例子。

1. 蠕虫

蠕虫应用程序可以创建一个确切的或类似的副本通过网络或可移动媒体传播。例如，蓝牙蠕虫可以利用蓝牙功能，并发送副本到配对的设备。Android.Obad.OS众所周知蓝牙蠕虫。

D.僵尸网络

僵尸网络应用程序妥协的在设备创建一个使该装置是由一个通过一系列的命令远程服务器控制的，称为BOT的主机。这种网络称为僵尸网络。命令可以将私人信息发送到远程服务器或是复杂的导致拒绝访问的服务攻击命令。Bot 还可以包括自动下载恶意的有效载荷的命令。Geinimi ，Anserverbot，Beanbot 是显著的Android僵尸网络。



E.间谍软件

间谍软件可能表面看起来是一个很好的实用程序，但它有一个隐藏功能，那就是暗中关注个人的联系人，短信，位置，银行MTANs等，这将导致巨大的不良后果。它可以将收集到的信息发送到远程服务器。Nickyspy，GPSSpy 是已知的间谍软件应用程序的例子。

F.攻击性广告

安卓提供粗、细粒度的两种精度不同的位置服务。一些广告子网络滥用这样的位置服务发送个性化广告给用户设备以产生收入。攻击性广告软件可以创建主屏幕上的快捷方式，窃取书签，改变默认的搜索引擎设置和推不必要的通知这阻碍了有效的设备使用。Plankton 是已知的攻击性广告。

G.勒索

勒索软件可以锁定用户设备使其无法访问，除非用户用在线支付的方式向其支付了赎金。例如， FakeDefender.B伪装成avast！反恶意软件，并显示假恶意软件提醒哄用户安装此恶意软件。然后，它锁定设备要求赎金解锁设备。

1. 恶意软件渗透和生存技术

在这一节中，我们总结了恶意软件的渗透和安卓系统应用于安卓的恶意软件应用程序技术状态。

1. 重新流行的应用

包装是一个从流行的应用市场分解/反编译流行的免费或付费应用程序，然后插入附加的恶意软件的有效载荷，重新组装的木马程序，并通过较少监控的本地应用程序商店分发它们的过程。一个应用程序用现有的逆向工程工具可以被重新包装。图6所示是包装过程。下面的部分讨论了应用程序的重新包装的主要步骤：

·从AppStore下载免费/付费应用。

·拆卸反汇编程序，例如apktool 。

·生成Java Dalvik字节码或恶意的有效载荷并将其转换为使用DX 工具的字节码。

·将恶意软件的有效载荷添加到良性应用程序。如果需要，修改AndroidManifest.xml 文件

·再次使用apktool组装修改的源代码。

·通过与另一个自签名分发打包应用程序证书到少监控第三方应用程序市场。

包装是一种最常见的恶意软件的应用程序生成技术。来自合法的官方应用市场的样本，其中80%的软件的基因组数据集被重新打包的恶意软件变种。重新包装技术可以用来大量产生恶意软件的变种。它也可以用来产生一些看不见的已知恶意软件的变种。作为每个恶意软件的变种的签名，商业反恶意软件检测看不见的恶意软件。包装是一个很大的威胁，因为它可以污染应用程序的市场，也伤害了第三方开发商的声誉。恶意软件作者可以通过更换广告来转移原来的开发商的广告收入。AndroRAT APK粘结剂重新包装并产生木马版本的流行的和合法的配备远程访问功能。对手可以远程强制被感染的设备发送短信，使语音呼叫，访问设备的位置，记录视频和/或音频和访问使用远程访问服务的设备文件。

1. 通过驱动下载

攻击者可以使用社会工程学，积极的广告，点击一个恶意网址，煽动用户自动下载恶意软件。可以选择，通过下载驱动可以伪装一个合法的应用程序，并哄用户安装应用程序。Android / NotCompatible是一个显著的驱动下载的应用程序。

1. 动态负载

一个应用程序也可以嵌入恶意负载作为一个可执行的apk /jar内的APK

格式加密或资源。一旦安装，这个程序便解密有效负载。如果有效载荷是一个JAR文件，恶意软件加载dexclassloader API执行动态代码。然而，它可以欺骗用户作为一种重要的更新安装伪装并嵌入式APK。应用程序可以执行本地二进制文件使用runtime.execAPI，Linux fork() / exec()等。BaseBridge 和Anserverbot 恶意软件族采用上述讨论的技术。些恶意软件族不嵌入恶意负载作为一种资源，但它们从远程服务器下载，并成功地逃避检测。DroidKungFuUpdate 是一个著名的动态执行有效载荷的例子。这种技术没有和静态分析方法一样被发现。

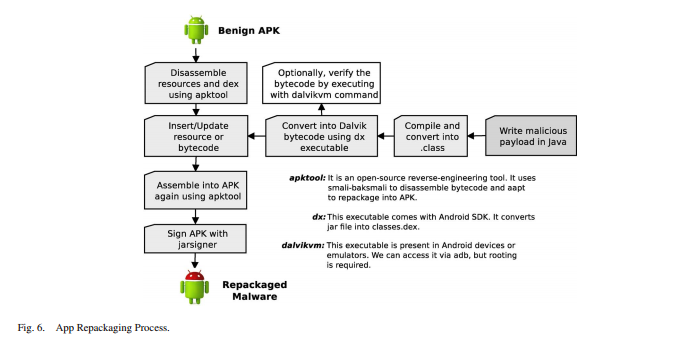
1. 隐身恶意软件技术

Android操作系统是为资源受限的环境开发的，这是一个在头脑中的可用性有限的基本智能手机的可用性。关于设备反恶意软件应用程序不能进行实时的深度分析，不像他们的桌面对应。恶意软件作者利用这些硬件限制反恶意软件的约束和混淆的恶意的有效载荷来阻挠商业反恶意软件。隐身技术，如代码加密，密钥排列，动态加载，反射代码和本机代码执行仍然保持基于签名的反恶意软件解决方案的问题。在桌面平台的趋势，代码混淆也是Android 发展。混淆技术实施的一个或多个以下用途。

·为保护专有算法的竞争对手通过逆向工程困难。

·为保护多媒体资源的数字版权管理以减少盗版。

·混淆应用程序使他们紧凑，从而更快执行。



·为隐藏从反恶意软件扫描仪扫描出的已知的恶意软件来传播和感染更多的设备。

·为了防止或至少延误人类分析或自动分析发动机，计算出未知的恶意软件的实际的动机。

Dalvik字节码进行逆向工程由于类型安全信息，例如，类，方法类型，定义，变量，寄存器的字符串和指令。在Dalvik字节码中代码转换方法可以很容易实现，优化与代码保护工具如混淆器。ProGuard是删除未使用的类的优化工具，方法和字段。有意义的类/方法/领域/局部变量的名字用不可读的代码替换，用硬化的逆向工程。Dexguard是一个商用安卓代码保护工具。为了保护应用程序的逆向工程，它可以用来实现代码混淆技术如加密类，合并的方法，字符串加密，控制流混淆等。代码转换技术也可以用来阻碍恶意软件的检测方法。Faruki等，推出了一个自动化Dalvik字节码转换框架为生成看不见的已知的恶意软件以及不同变体的字节码混淆技术。此外，他们还评估了看不见的恶意软件样本对顶部的商业反恶意软件和静态分析技术。作者报告说，即使是微不足道的转换技术也可以使现有商业软件失败。在下面，我们覆盖各种代码转换方法混淆现有已知的恶意软件和生成大量看不见的恶意软件签名。事实上，为阻止拆卸工具，代码也可以实现改造。

1. 垃圾代码插入和操作码排序：垃圾代码或无操作码（NOP）的插入是一个众所周知的改变文件的大小和逃避反恶意软件签名数据库的技术。垃圾代码插入保留了原始应用程序的语义。然而，它改变了操作码序列来改变恶意软件应用程序的签名。操作码可以重新排序用goto指令之间的相互作用和改变控制流，保留原始执行语义。这些方法可以用来逃避或基于opcodebased检测解决方案的签名。
2. 包、类或方法重命名：Android应用程序独特的包装名称唯一标识。Dalvik字节码是类型安全的保存类和方法的名字。许多反恶意软件使用平凡的签名，如包，类或方法名称的一个已知的恶意软件作为检测签名。这些琐碎的变换可以用来逃避检测反恶意软件的签名。
3. 改变控制流：一些反恶意软件的使用语义如控制流和/或数据流分析的签名用简单的变换检测恶意软件的变种技术。用GOTO指令，或通过插入和调用垃圾代码的方法可以修改程序的控制流。虽然琐碎，但是这样的技术可以回避商业反恶意软件。
4. 字符串加密：文字字符串，如消息，网址和shell命令揭示了很多关于应用程序。为了防止这样的分析，可以加密的纯文本字符串使其无法读取。此外，每次执行串加密，各种加密方法（或键）使其难以自动化解密过程。在这种情况下，文字串只有在代码执行期间是可用的。因此，回避静态分析的方法。
5. 类加密：通过利用上述敏感加密整个类信息，重要信息，如产品许可证，检查，付费下载和DRM可以隐藏。
6. 资源加密：资源文件夹的内容，资产并且本机库可以被改变为不可读的，因此他们在运行时必须解密
7. 使用反射的API：静态分析方法搜索该恶意软件的应用程序中的敏感的Android API映射的恶意行为。用户的应用程序允许Java反射允许创建方案类实例或方法调用使用文本字符串。要确定确切类或方法名，数据流分析可以实施。

然而，文字串可以加密，使之硬自动搜索反射API。这样的技术可以很容易逃避静态分析方法。

1. 评估方法分析和检测

安卓安全解决方案，如漏洞评估，恶意软件分析和检测技术分为：1）动态2）静态3）混合动力。静态分析方法分析代码没有实际运行，因此它们很快，但他们必须处理假阳性。动态分析技术监控执行代码，检查其相互作用与系统。虽然费时，但是它们对恶意软件的变种是有效的。混合方法合理的利用静态和动态分析方法。

安全解决方案可以被分类为以规则为基础的或基于特征提取的机器学习模型。不恰当的特征选择可以降低模型的性能，产生假阳性（即，假检测的良性作为恶意软件的应用程序）。此外，问题的特征数目必须是小而有效的作为一个在设备上反恶意软件解决方案。特征降维方法统计的措施，如均值、标准差、卡方、Haar变换可以用来识别突出负责恶意行为的属性。学习模型可以通过分析处理器的特性来创建，内存使用，电池消耗，系统调用调用，网络活动等，可用于聚类或预测异常行为的分类算法。

1. 静态分析方法

基于静态分析的方法只可拆卸，反编译没有实际运行，因此不会感染装置。这种方法被各种各样的使用破坏了，在本文中的第5节第D小结中讨论代码转换技术.

1. 基于签名的恶意软件检测：现有的商业反恶意软件使用基于签名的恶意软件检测方法。它提取有趣的语法，语义，模式或特征，并创建一个独特的签名匹配那个特殊的恶意软件。基于签名的方法对于已存在的和已知的恶意软件的不可见的变种是无效的。此外，签名提取过程是手动的，其指数独特的签名爆发后的疗效可能离开该设备容易受到恶意软件攻击。Faruki等人推出了androsimilar方法，一个自动稳健基于统计特征签名的零日检测已知恶意软件的变种的方法。
2. 基于组件的分析：为了进行详细的应用程序安全评估或分析，应用程序可以被拆卸来提取重要的内容，如AndroidManifest.xml，资源和字节码。AndroidManifest.xml存储重要的元数据，例如组件列表（例如，活动、服务、接收器等）和所需的权限。应用程序的安全性和评估解决方案可以用自己的定义分析，利用字节码交互组件确定漏洞。
3. 基于权限的分析：请求权限访问敏感资源是安卓系统的核心设计安全模型。没有应用程序可以默认的利用任何权限而影响用户的安全。识别危险权限请求不足以声明该恶意程序应用程序，但尽管如此，权限映射请求和使用权限是一种重要的风险识别技术。Sanz Borja等人利用AndroidManifest.xml中的uses− permission和uses− features标签来识别恶意软件。作者运用机器学习算法的朴素贝叶斯，随机森林，J48和贝叶斯网络在249恶意软件和357个良性应用程序集。在作者映射的要求和使用权限清单和相应的API在dalvik字节码。这个映射属性被用来与机器学习算法的125249个恶意软件和良性的应用程序集。恩克等人。开发了一套认证工具，麒麟定义了一套规则确定特定危险权限的组合在安装应用程序之前识别恶意软件属性。
4. Dalvik字节码分析：Dalvik字节码是语义丰富的包含类型信息如类，方法和指令。类型信息可用于验证应用程序行为。基于控制和数据的详细分析流动提供了一个洞察危险的功能，如隐私泄漏和电话服务滥用。控制和数据流分析，也有助于重建德混淆字节码，例如合作效应琐碎变换技术。

字节码的控制流分析确定一个应用程序可以同时执行的可能的路径。Dalvik字节码包含跳跃、分支和方法调用指令改变执行顺序。为了便于进一步分析，术中（即，在一个单一的方法或程序间）（即，跨越方法）控制流的字节码图（CFG）产生。卡尔森等人。 正式在Dalvik字节码以执行基于语义控制流分析签名来检测恶意软件的应用程序。

字节码的数据流分析预测了尽可能多的不同的执行点时的值。CFG可用于遍历可能的执行路径，以确定控制和数据相关性。执行数据流分析内的方法（过程内）或不同的方法之间（过程间级），提高逼近所需的输出。特别是，特殊数据流分析也被称为“传递”的实施，以确定在调用敏感API调用不断的争论应用程序的执行。例如，一个恶意的应用程序发送收费可以检测率SMS来预先定义的硬编码号码随着不断的传播数据流分析。污点分析中的其他类型的数据流分析的方法来识别着色变量保持的敏感信息。例如，污点分析可以识别的隐私泄漏其可以是用于窃取敏感的用户信息的应用程序。敏感API调用字节码内跟踪可能是有用的，以识别恶意行为。它也是在确定程序有助于克隆。 Zhou等人。利用操作码的序列在Dalvik字节码指令，以确定重新包装恶意软件的应用程序。

1. 将Dalvik字节码重新定位为Java字节码：基于Java的数目的反编译和根据分析工具的静态分析的可用性，已促使研究人员重新瞄准Dalvik字节码转为Java字节码。恩克等开发了用于转换Dalvik字节码到Java源代码的DED工具。后来，他们进行静分析控制流，数据流，使用Fortify的Java代码SCA的框架。在作者开发的工具中Dalvik字节码转换为Java字节码具有99%的精度。巴特尔等人 开发的Dexpler插件是基于静态分析框架的。Dexpler转换的Dalvik字节码转换为煤烟的内部Jimple代码。然而，它无法处理DEX（ODEX）文件的优化。Gibler等。采用DED和dex2jar将Dalvik的字节码分别转换为Java字节码和源代码。作者实现静态分析WALA框架识别一个相当大的Android应用程序中的隐私泄露数据集。
2. 动态分析

静态分析和检测方法是快速的，但是它们对加密，多态转化代码恶意软件是无效的。动态分析方法在受保护的环境下执行应用程序，提供它需要的所有的模拟资源，以便了解它的交互识别恶意活动。一些动态的分析方法已经实施，但智能手机有限的资源约束这种执行方法。Android应用程序的执行是事件基于异步多个入口点，重要的是触发的事件。用户界面（UI）手势，如敲，捏，轻扫，键盘和背面/菜单键按一定自动触发启动与应用程序交互设备。Android SDK中自带配备了monkey 工具，自动执行某些以上讨论上述手势。为了进行了深入的监测，可以需要通过将称为跟踪代码修改框架仪表。

动态方法的一个严重的缺点是，如果它是根据触发一些不平凡的事件，一些恶意的执行路径可能会错过。例如，在特定的时间执行恶意软件功能的一天，但这种情况下永远不会执行。如沙盒反模拟技术检测，超时的分析环境，延迟执行的恶意软件能够回避动态分析的方法。动态方法分为以下三大类。

1. 基于轮廓的异常检测：恶意应用程序可通过在利用创建拒绝服务（DoS）攻击受限的硬件资源。参数范围CPU使用率，内存使用率统计，网络流量模式，电池的使用和系统调用的良性和恶意软件都是应用程序从Android的子系统收集的。随着机器学习方法自动分析技术用于区分异常行为。
2. 恶意行为检测：具体的恶意行为，如敏感数据泄露，发送短信/电子邮件，未经用户许可的语音呼叫，可以通过监测所关注的特定的特征精确地检测。
3. 虚拟机自省：对应用程序行为监控，从仿真器的缺点（VM），仿真器本身可能对恶意应用程序，它失败分析的目的。为了解决这个问题，虚拟机自省方法可以采用通过检测的应用程序的行为观察活动了模拟器的。
4. 部署评估，分析，与检测研究

安全评估，恶意软件分析和检测方法可以部署在不同的位置，这取决于从设备上的解决方案，以一个完全脱落装置或云基础技术的要求。

1. 在设备上

基于签名的恶意软件是简单而有效的。详细的评估和分析相比于桌面反恶意软件分析仍然受到移动的限制。从而，轻巧的风险评估解决方案可议通过分析组件和权限，作为对设备解决方案以下是一些在设备上的反恶意软件的限制。

•无需任何特殊权限运行作为一个正常的应用程序，反恶意软件应用程序。这样一来，他们也是权限下的进程隔离。因此，它们不能直接扫描其他应用程序的内存，文件读取/在书面和私人文件应用程序扫描。

•安卓允许执行的后台程序服务。但是，它可以停止运行反恶意软件应用服务，如果它对于资源的需求超出硬件资源。类似特权的应用程序可以强行停止反恶意软件程序执行适当的特权。

•如果没有获取root权限，反恶意软件应用程序不能创建系统挂钩监视文件系统或进行网络访问。

•如果没有获得root权限，反恶意软件应用程序不能卸载任何其他应用程序。它必须取决于用户是否要去除该应用。

1. 分发（某些型号的设备上，有部分关闭设备）

在飞行中分析和/或检测可以执行上设备，详细的计算成本分析可以在远程服务器上进行，使反恶意软件程序有限的资源友好。在这样的情况下，基于轮廓的异常检测，资源使用参数是在被收集客户端和发送回远程服务器以进行详细分析。结果可以最后发送回设备。然而，互联网带宽的持续可用性和相关的成本是一个问题。如遇网络资源的基于主机的检测方法可以保护设备不可用从恶意软件攻击。

1. 脱离装置

对于新的恶意软件样本自动的深层静态分析是非常重要的，以使人类分析师迅速采取决定旨在发现和减少恶意软件。这种自动化深入分析解决方案需要计算机的计算能力和内存。由于这个原因，它们通常被部署脱离装置。

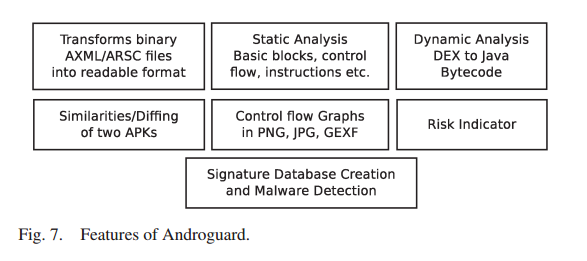
1. 最先进的安卓应用程序的评估，分析和检测的工具和技术

工业界和学术界已经对Android的恶意软件分析和检测提出了几种解决方案。在本节中，我们调查和研究有前途的逆向工程工具和检测方法。检测方法根据下列条件已被归类：1）目标，它可以是应用程序的安全性评估，分析或恶意代码检测; 2）方法在第六节讨论;3）部署讨论在第七节。

A.逆向工程工具

Android包（APK）的内容被存储为二进制格式。前评估，分析或检测任务的发起，拆卸它进行进一步的处理是很重要的。有一些工具拆卸和/或反编译了Android应用程序。在下面的章节中，我们讨论了一些已知的逆向工程工具，考虑到了它们的优点。

1. apktool可以将APK的二进制内容解码转换成在项目类的目录结构接近原来的形式。它反汇编二进制资源和字节码转换中classes.dex到smali字节码为方便阅读和操作。使得变化后，它也可以重新打包它放回一个APK。本工具是最好的开源逆向工程工具之一。
2. dex2jar是一个反汇编器，用来解析.dex和优化的DEX文件，提供一个重量轻的API来访问它。 dex2jar也可以转换成DEX JAR文件，由重新定位在Dalvik字节码转换为Java字节码，用于进一步的操作。此外，它也可重新装配jar文件为修改后.dex。
3. Dare项目的目的是重新定位Dalvik字节码在classes.dex传统的.class文件使用强大的类型推断的算法。.class文件就可以通过一系列的Java应用程序的开发的传统技术，包括反编译被进一步分析。 Octeau等。表明，Dare为40％比dex2jar更准确。
4. Dedexer拆开classes.dex成Jasmin的语法，并为每个类创建一个单独的文件，方便维护阅读和操纵包目录结构。然而，apktool不同，它不能重新组装DIS组装中间类文件。



JEB是一家专业的Android逆向工程，可在Windows，Linux和Macintosh平台使用。它是一个基于GUI的交互式反编译工具，可分析逆转恶意软件应用程序的内容。应用程序的信息，如manifest，资源，证书，文字字符串可以通过交叉引用提供了一个方便的进行审查的Java源代码。JEB通过使用Dalvik字节码的语义直接将Dalvik字节码转换为Java源代码。在特殊情况下，JEB还可以去混淆Dalvik字节码，相比同行使得反汇编代码的可读性更高。JEB支持Python脚本或插件允许反编译Java代码通过API摘要访问语法树（AST）。在自动化定制的分析此功能非常有用。据我们来说，这是迄今为止最好的逆向工程工具。

B. Androguard

目标：风险评估，分析和检测

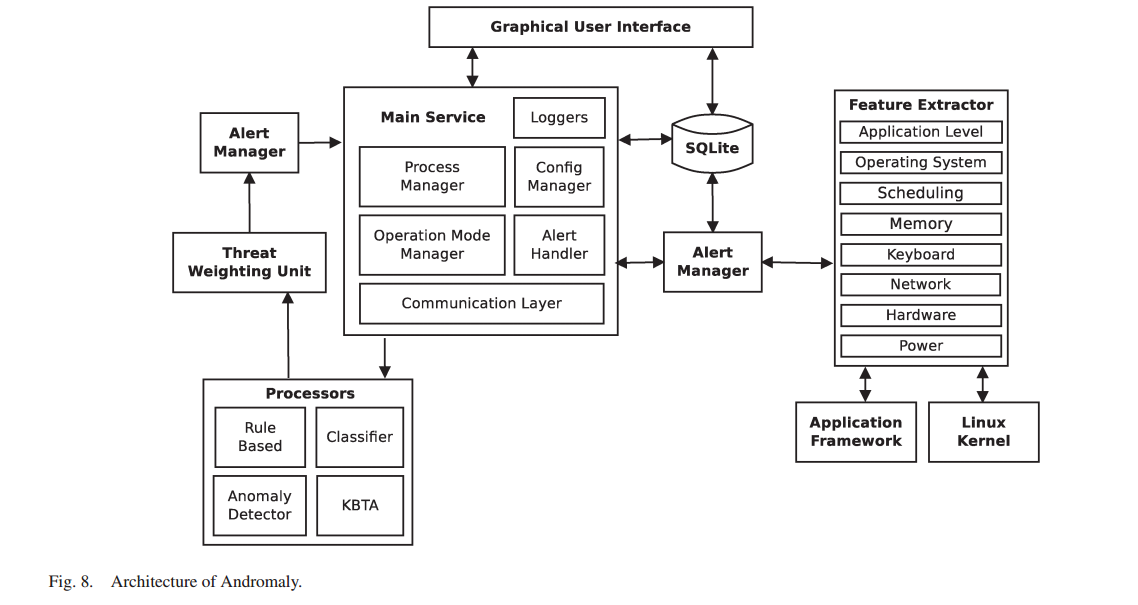
方法：静态

部署：脱离设备

见图7，Androguard是一个开源的，可以反向工程拆解和反编译Android应用程序的静态分析工具。它对每一种方法产生控制流图，并通过对Python的API提供命令行和图形界面。Androguard归压缩距离（NCD）的方法发现的相似性和

对两名涉嫌克隆可靠，这也是差异有助于发现重新打包的应用程序。它提供了Python的API来访问拆卸资源和静态分析结构如基本块，控制流和APK的指令。一个分析师可以通过利用Python API开发自己的静态分析框架。以下是一些功能介绍。

1. 应用程序代码相似：Androguard通过计算标准化压缩每个方法对和之间的距离计算0-100之间的相似性得分计算两个应用程序之间的相似性，其中100是指相同的应用程序。它显示相同，相似，新建，删除，跳过了两名涉嫌克隆的方法。在相同的方式下，它显示通过比较两种方法之间每个基本块的差异。更具体地说，为了计算两个类似的方法之间的差异，它首先将每个独特指令在基本块转换为一个字符串。然后，它应用于最长公共子算法上的两个基本这些字符串块，以查找它们之间的差异。



1. 风险指标：风险指标模糊计算APK从0（低风险）到100（高风险）的风险评分。它考虑以下参数：

•本机，反射，密码和应用程序中的动态代码。

•应用程序中的可执行文件或共享库的数量。

•涉及到隐私和财务风险的权限请求

•其他系统危险或签名许可请求。

1. 恶意应用程序签名：Androguard管理着签名数据库，并提供一个接口用来从数据库中添加或移除签名。签名用JSON格式描述。它包含一个名称（或家族名称），设置子签名和一个布尔公式混合不同子签名。以下是两种类型的子签名： •METHSIM：它包含三个参数，CN级命名，MN-方法名和D-描述符。

•CLASSSIM：它包含了一个参数，CN级名称。

因而子签名可以在一个具体的方法或整个类中被应用。不同的子签名可以混合

布尔公式（BF）。

C. Andromaly

目标：异常检测

方法：动态

部署：半设备，半脱设备

Shabtai等。提出了一种轻量级的Android基于机器学习的恶意软件检测系统。它实时监控收集各种系统参数，如CPU使用率，数据通过网络传输量，活动进程和电池的使用情况。如图8，Andromaly有四个主要组成部分：

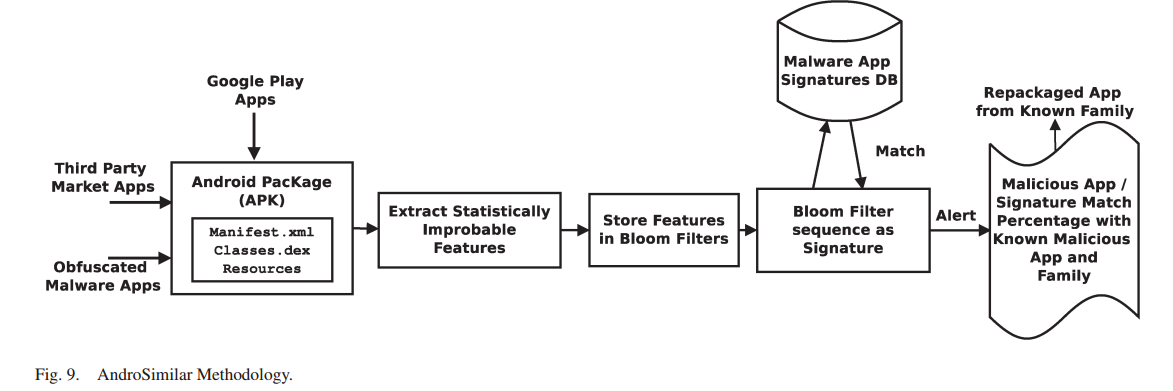
•特征提取：他们收集功能的指标，通过和Android内核和应用程序框架进行通信。特征提取器以有规律的触发间隔的功能管理器收集新功能的测量。功能管理器还可以执行一些对原始特征数据预处理。

•处理器：这是一个分析和检测装置。它接收来自主服务的特征向量，分析

他们并进行威胁评估，并把它传给威胁加权单元（TWU）。处理器可以基于规则的，以知识为基础的分类或异常探测器采用机器学习的方法。 TWU应用集成算法从接收到的结果的分析所有的处理器来导出装置上的最后决定感染。警报管理器平滑结果，以减少误报。

•主服务：它负责协调要素集合，恶意软件检测和报警处理。它负责请求新功能测试，发送新功能的指标来处理器和接收来自最后建议警报管理。记录仪可以记录的信息进行调试，校准和试验。 组态管理器管理的应用程序的配置，例如，活动处理器，警报阈值，采样间隔等激活或停用处理器的任务是照顾由处理器管理器。 操作模式经理切换应用程序从一个模式到另一个其导致处理器的激活/去激活和特征提取。这种变化在操作模式是由于造成资源水平改变

•图形用户界面：它与用户交互配置应用程序参数，启动/关闭应用程序，提醒有关用户的威胁，并允许探索收集的数据。实验进行了使用几类人工恶意软件，因此工作模式需要测试通过真正的恶意软件。



D. AndroSimilar

目标：恶意软件检测

方法：静态

部署：脱离设备（也可移植到设备上）

文献作者提出AndroSimilar，提取统计上罕见的语法自动签名生成方法功能恶意软件检测。除了现有的恶意软件，AndroSimilar能够合理地检测恶意软件的混淆与像字符串加密，方法重命名，垃圾技术方法插入和变化控制流程，广泛用于躲避固定的反恶意软件签名，因此，它可以检测未知现有的恶意软件的变体。 AndroSimilar方法是基于相似度摘要哈希（SDHash）应用于数字取证，以确定类似的文件。直观地说，完全无关的应用程序应该具有较低概率有共同的特点。当两个不相关的应用程序共享一些功能，这样的功能，应考虑弱使用这些将导致误报。固定大小的字节序列的特征是基于经验提取的出现它们的熵值的概率，当时流行的功能是搜索其中根据专挑居委会。 图9显示了AndroSimilar的工作过程。以下是涉及的步骤：

•提交Google Play，第三方或混淆的恶意应用程序作为输入输入AndroSimilar。

•根据文件和规范，为每一个固定大小的字节序列生成熵值，这些值在[0，1000]范围内。

•根据相似性选择统计强大的功能摘要计划为代表的应用程序。

•将提取的特征存储到布鲁姆过滤器。序列布鲁姆过滤器是一个应用程序的签名。

•与数据库中已知的恶意软件系列进行匹配检测比较签名。如果相似度得分超出给定的阈值，将其标记为恶意（或重新打包）样品。

因此，它们产生的已知恶意软件家族作为签名具有代表性的数据库。如果一个未知的应用程序相似性得分与任何现有的家庭特征相匹配超出阈值，然后它被标记为恶意的。我们相信AndroSimilar是有前景的检测看不见的恶意软件变种的方法。

E. Andrubis

目标：恶意软件分析和检测

方法：静态和动态

部署：脱离设备

Andrubis是一个基于网络的恶意软件分析平台，在一些现有的知名工具中内置Droidbox，TaintDroid，apktool和Androguard。用户可以通过基于Web的界面提交可疑的应用程序。在远程服务器上的应用程序分析后，Andrubis返回详细的静态和动态分析报告，作为一个网页。 Andrubis还提供了应用程序之间的行为评定0-10，其中0表示良性和10指定的恶意等级。为研究Andrubis的功能，定制的基于SMS的僵尸网络被上传到Andrubis Web服务。这项研究原型额定定制短信机器人，得分9.9/10。然而，没有门户商业反恶意软件能够探测到这种看不见的恶意软件。这说明了Andrubis行为评定反对的有效性零日恶意软件。然而，维达斯等，展示该Andrubis虚拟环境中检测与抗分析技术鉴定分析沙箱。

F. APKInspector

目标：恶意软件分析

方法：静态

部署：脱离设备

APKInspector是一个全面的Android静态分析工具，包括扣减smali/ baksmali，apktool和Androguard。它提供了丰富的图形用户界面，并具有以下

产品特点：

•应用程序的元数据

•敏感权限的分析

•显示Dalvik字节码和Java源代码

•显示控制流图

•显示调用图，显示呼入和呼出结构

•静态工具的支持，允许修改该smali代码

G. Aurasium

目标：分析与检测

方法：动态

部署：设备

Aurasium是一个强大的技术，采用的控制执行的应用程序，通过强制执行任意的运行安全策略。为了能够做到这一点，Aurasium重新包装了Android

应用程序的策略执行模块。Aurasium安全管理器组件可以对个人适用的政策和多个应用程序。任何安全和隐私侵犯报告给用户。因此，它不需要用于操纵

Android操作系统监视应用程序的行为。它介入，案件应用程序访问敏感信息，如联系人，短信，电话标识符和执行shell，命令由要求用户输入就同一确认。Aurasium由它屈从于隐身的事实限制恶意软件，也就是说，它可以通过应用程序基于签名检测修改和预定义本机库的存在。恶意软件

应用程序可能无法揭示其恶意行为，如果它标识Aurasium的存在，从而避免了检测。由于Aurasium取决于重新包装，它可能无法拆卸（或组装）转换代码的应用程序。

H.Bouncer

目标：恶意软件检测

方法：动态

部署：脱离设备

谷歌用称为Bouncer的系统保护其自己的应用商店，Google Play。它是一个基于虚拟机的动态分析平台来测试上传的第三方开发应用程序，在它们被用户下载之前。它执行应用程序，以寻找任何恶意行为，并进行比较针对前面所分析的恶意应用程序。虽然内部运作没有文件可用，Oberheide等。提出了他们的Bouncer环境分析，通过实现自定义命令和控制应用程序。动态代码加载技术可以逃避Bouncer的审查。

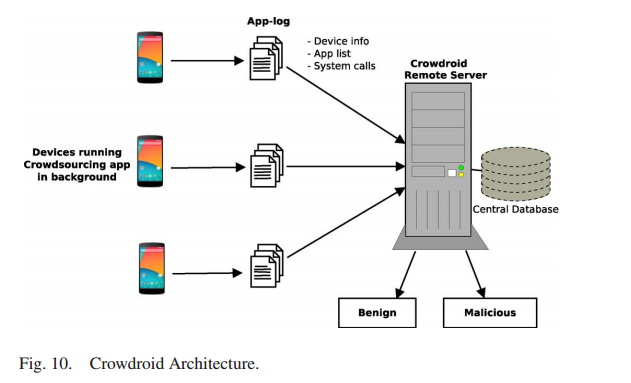
I.CopperDroid

目标：恶意软件分析和检测

方法：动态

部署：脱离设备

雷纳等人。提出CopperDroid，一种执行Android应用程序的系统调用为中心的动态分析，使用虚拟机的反思的系统。为了解决路径覆盖问题，他们已经支持按照目前的应用程序清单文件中的规范。作者已经通过实验证明，系统呼叫中心的分析，可以有效地检测恶意行为。他们还提供了一个Web界面为其他用户分析应用程序。然而，Vidas 等。 通过使用证明CopperDroid的虚拟环境的识别先进的防分析技术。



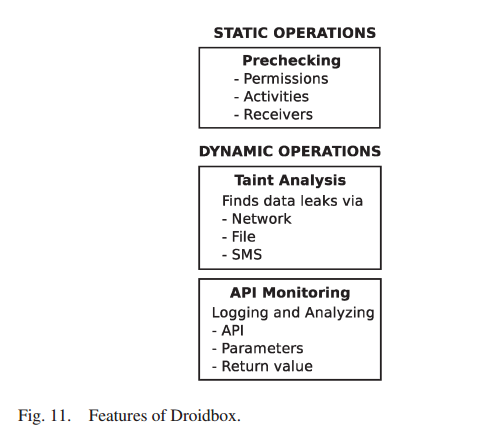
J. Crowdroid

目标：恶意软件检测

方法：动态

部署：半设备上，半断设备

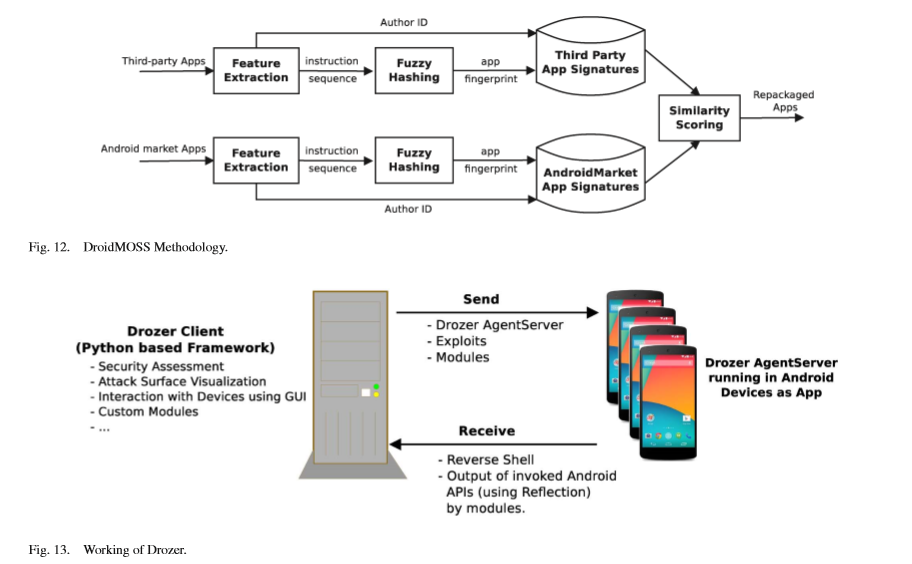
Crowdroid是基于行为的恶意软件检测系统（参照图10）。它由两部分组成，一大群采购被安装在用户的设备和用于RemoteServer的恶意软件检测应用程序。人群采购应用程序发送行为数据（即系统调用的详细信息）的形式一个应用程序的日志文件到远程服务器。 strace的，一个系统在设备工具本是用来收集系统，呼叫详细信息的应用程序。应用程序日志文件由基本的设备信息，安装的应用程序和行为数据的列表。在远程服务器上，该数据被处理以创建特征载体可能然后通过分析2，意味着分区聚类预测应用程序为良性或恶意的。一个应用报告生成并存储在遥控器的数据库服务器。



Crowdroid的结果是准确的自我编写的恶意软件有为一些真正的恶意软件。如果恶意软件是非常活跃，则有可能在系统差大电话，其可在检测同一帮助。但是，它也遭罪假阳性，就证明了使用的作者猴子Jump2，与HongTouTou恶意软件的应用程序。

Crowdroid的限制，人群采购应用程序必须的始终可用以监测，这可能会耗尽可用设备资源。此外，这种技术还有待检验可用来确定已知的恶意软件家族的

效力。



K. Droidbox目标：污点分析和监测

方法：动态

部署：脱离设备

Droidbox如图11所示，是一个基于TaintDroid之上开发的动态分析工具。它修改了Android的框架为API调用分析。 图11显示了Droidbox的静态和动态分析操作。应用分析始于静电预检，其中包括解析权限，活动和接收器。应用程序被分析时是在仿真环境中执行进行污点分析和API监控。污点分析涉及通过程序变量，文件和进程间通信（污点）传播的私人和敏感数据的标签。污点分析跟踪被污染数据的离开系统无论是通过网络，文件或短信，并发送应用程序负责过滤。API监测涉及API记录其参数和返回值。 结果包括以下参数：

•应用程序的哈希值

•网络数据传输或接收

•文件读写操作

•数据泄漏

•规避权限

•广播接收器

•启动服务，并通过DexClassLoader加载类

•短信发送和已拨电话

•与Android API实现加密操作

•时空运营秩序

•树图的相似性分析

限制：Droidbox只能监控在Android框架中执行的任务。如果本机代码漏敏感数据，现有的系统不能检测，因此数据是在用户不知情的情况下前过滤。

L. DroidMOSS

目标：重新包装应用程序检测

方法：静态

部署：脱离设备

DroidMOSS是一个采用语义文件相似的措施重新包装检测原型的应用程序。 进一步来说，它提取一个应用程序的DEX操作码从操作码的签名序列生成签名模糊哈希。它还增加了开发者证书信息，映射成签名唯一的32位标识符。可疑的应用程序的功能使用编辑距离，对原来的应用程序进行验证算法来识别相似性得分。建议的方法讨论并在图12所示。后面DroidMOSS直觉使用操作码的特点是，它也许是容易的对手进行修改操作数，但很辛苦改变实际的操作码。这种方法有几个缺点。首先，只考虑DEX字节码，无视本机代码和应用程序资源。二，操作码序列做不是由高层语义信息，从而产生假阴性。聪明的对手可以轻易逃避这种技术使用如插入垃圾代码转换技术字节码，重组方式和旧的控制流量，以逃避该DroidMOSS原型。

M. DroidScope

目标：分析

方法：动态

部署：脱离设备

DroidScope是一个基于动态分析的虚拟机自省（VMI）Android框架。不像其他动态分析平台，它保持了仿真器，监控操作系统和Dalvik的语义。因此，即使是特权攻击Android内核，它也可以删除它。它也使得破坏攻击的分析任务变得困难。DroidScope是建立在QEMU仿真器与丰富的API来定制恶意软件分析的原型。安卓恶意软件系列DroidKungFu和DroidDream被这种技术进行了分析与检测。然而，DroidScope对其他恶意软件系列的有效性还有待测试。

N. Drozer

目标：通过开发风险评估

方法：静态和动态

部署：半设备，半脱离设备

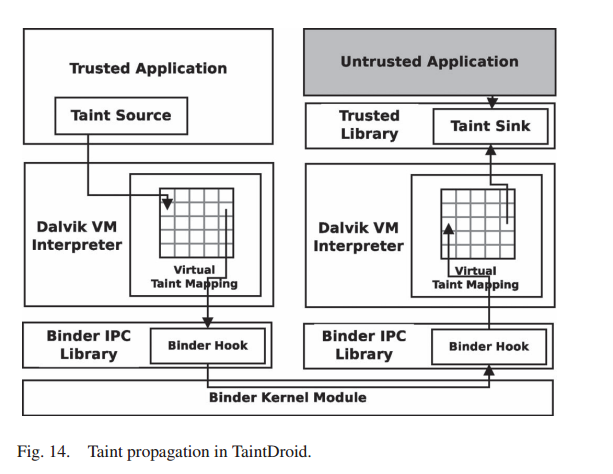
Drozer是一个针对Android设备的综合性的攻击和安全评估框架，开放源代码和专业版。它可以让安全执法机构以远程方式利用Android设备识别Android操作系统的漏洞和威胁。图13显示了Drozer的功能。以下是Drozer支持的功能列表：

•它安装一个可以利用Java反射API执行开发模块的代理app。在服务器端，用Python可以创建自定义模块并将其发送到代理程序来执行开采活动上的设备。

•它可与Dalvik虚拟机进行交互，发现安装封装及相关应用程序的组件。它还允许与应用程序，组件，如服务的交互，内容供应商和广播接收机，以找出安全漏洞。

•它可以创建一个shell与Android操作系统进行远程交互。

•它能够产生已知漏洞采取已知生根漏洞的优势。



O.Kirin

目标：风险评估

方法：静态

部署：设备上

作者提出一种安全策略实施机制Kirin，一个设备上的应用程序，审批框架。Kirin定义了由应用程序要求的基于一定的组合组规则的危险的权限。如果一个应用程序不满足Kirin的安全规则，安装将被阻止。因此，该方法做出决定的基础上的一套规则，对用户代表。

P. TaintDroid

目标：污点分析

方法：动态和Android仪器

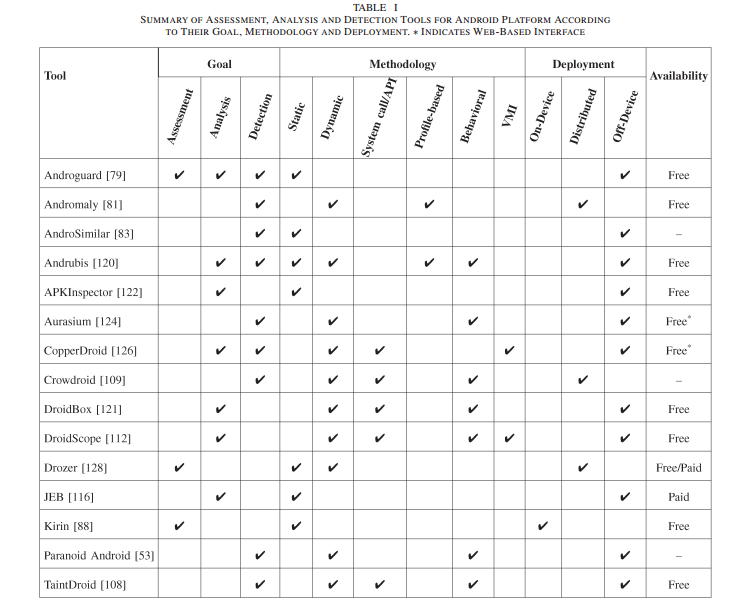
部署：脱离设备

TaintDroi为了跟踪在第三方开发商的应用程序的隐私敏感信息泄露，扩展了Android平台。标记的数据是否离开敏感数据自动污点（或标记），以便跟踪

设备。当敏感数据离开系统，TaintDroid记录的特定数据的标签，并发送该应用程序的连同其目标地址中的数据。污点传播跟踪在四个级别的粒度，1）可变的水平，2）方法级，3）消息级4）文件级。可变级跟踪使用可变语义，它提供了必要的条件，以避免感染传播。在消息级别的跟踪，对信息的污点被跟踪，以避免IPC开销。方法级跟踪用于Android原生不能直接访问应用程序，而是通过库修改固件。最后文件级跟踪确保诚信文件访问活动，通过检查污点标记是否保留。

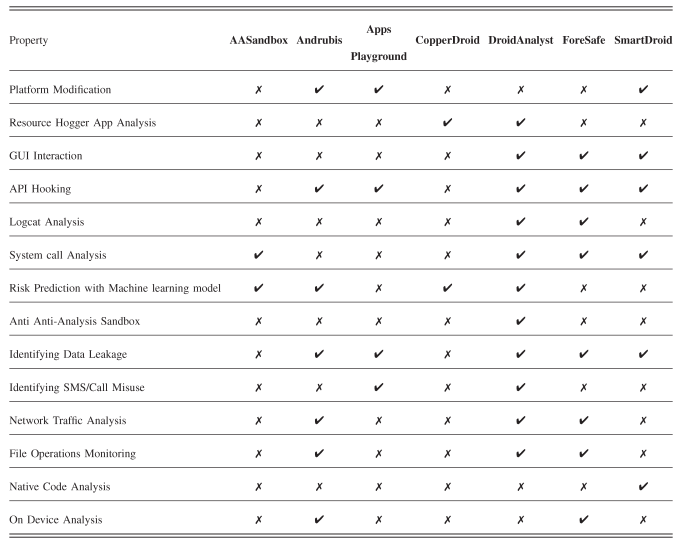
让我们考虑TaintDroid的工作中，其中数据一个可信的应用程序是由一些不信任的应用程序访问和发送在网络上。上述情况下，显示在图14。首先，受信任的应用程序的信息，根据被标记它的背景。与Dalvik虚拟机本地方法接口

解释器来存储污点标记在一个虚拟的污点地图。每个译员同时根据数据流规则传播污点的标签。该TaintDroid的粘合剂库被修改，以确保信任的应用程序的被污染数据被作为有污点标签反映了联合包裹所有的污点标记包含的数据。内核传送该包裹透明地在达到粘合剂库实例不受信任的应用程序。污点标签从包裹检索和标记为粘结剂库实例中的所有包含的数据。的Dalvik字节码解释器转发这些污点标签和对不受信任的应用程序组件请求的数据。当这应用程序调用污点宿（例如，网络）库，它检索污点标签和标志的行为为恶意。



Q.其他具有前景的技术

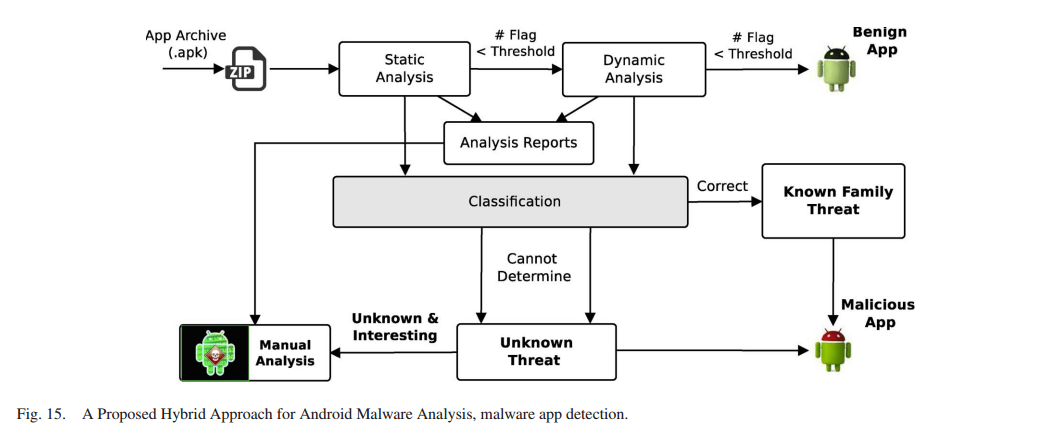
第三方应用开发商赚取的收入由免费软件使用应用程序内的广告库。一些广告机构提供的广告库由应用程序开发列入应用程序来赚取有针对性的广告的收入。AdRisk发现了一些攻击性的广告，这些广告利用用户的隐私进行有针对性的广告推广。已经有广告联盟的实例，分类越来越可疑，由于无论是有针对性的广告夹杂物或发送恶意广告，危及用户安全。因此，同样重要的是要在一个应用程序检测到这样的广告库，以作出明智的决定。AdDetect是一种比现有的方法更有前途的检测在应用程序内的广告库中合理的存在精度语义的方法。Damopoulos等提出的组合，并且基于云的入侵检测系统（IDS）。尤其是，作者强调了这样一个系统的重要性，当智能手机的网络资源的可用性低的时候为保护有限的网络资源它执行基于主机的检测。在装置电池耗尽时，原型智能地选择采用基于云的检测，利用云端的无限的处理和存储器。作者为视障人士在各种灯光条件下提出了一种室内导航。建议原型PERCEPT-V，智能手机基于UI采用了直观的标签以采样算法不同环境，照明，环境和使用的角度。



Vidas 等提出了一种系统的基础上的行为差异来确定模拟的Android环境，绩效评估，智能手机的硬件和基于功能的软件功能的存在/不存在。这样的系统采用突出的抗抗分析的重要性沙盘环境中的技术。Faruki等。提出了一种与平台无关的防反仿真沙盘探测隐形的Android恶意软件。作者还建议机器学习模型来预测资源hoggers。此外，作者提出了基于一种新颖的解决方案上的行为触发随机模型来检测目标，与先进的恶意软件。

能够将消息发送到保险费率数字的短信木马的增长是最大化的货币收益方式。Elish等设计了一个静态的异常检测方法以确定用户输入和回调到敏感的功能的参数之间的非法数据的依赖。使用这种方法，他们所表现出来的一些Android的恶意软件检测的用户不知情或不同意的发送邮件。然而，他们的做法没有考虑到帐户异步的API的Android如组件间的通信，从而未能检测复杂的短信木马，如Dendroid 。AsDroid是另一个有趣的静态分析工具，通过寻找之间的语义匹配检测隐形的行为用户界面的文本和敏感及其相应的使用特征。

Portokalidis等提出了另一种脱装置通过克隆智能手机的状态下远程服务器的恶意软件检测方法。远程服务器可以有很高的运算能力，更多的内存和同时不间断执行多种检测技术的电源。所提出的原型是可扩展的，实用的，并会导致较低的网络开销。在作者提出的比较框架不同动态分析，沙盒，以确定其中的局限性已知的基于Web的恶意软件自动分析框架。作者的结论是，现有的沙箱方法无法针对先进的和有针对性的恶意软件。然而，Faruki等提出了一种透明的，可扩展的Android版本不变的先进的沙箱检测Android威胁技术。



Rattazi等提出，标识了访问控制不存在或重要地区的系统方法不正确识别决策的主体与客体。作者进行了具体的实验来检验他们的假设得出的结论是较新的功能，还需要成熟作为移动互联网设备共享环境。Petsas等展示先进的恶意软件应用程序阻碍虚拟/模拟环境阻碍动态分析。作者修补现有恶意软件的应用与反分析功能来证明大多数的弱点在已经讨论过的现有框架。然而，Faruki等。提出了一个Android平台不变Android的沙盒揭开高级恶意软件失败的上述现有的基于Web的分析沙盘讨论。

R.基于网络的分析沙盒比较

在这里，我们讨论了沙盒的原型实现，作为Web服务，在Android的恶意软件呈指数增长的时代，以帮助人类分析师跟上时代的步伐。Andrubis和Copperdroid都在Taintdroid和Droidbox的基础上实现，隐私泄漏检测方法和动态分析技术。表二比较了各种网络根据沙盒办法被研究人员测试和自动恶意软件分析。手机沙盒是一个自动化恶意软件分析和检测方法结合本地代码分析，不提供与现有的原型的设施。Droidanalyst是一种抗反分析，沙盒来检测环境无恶意软件。应用程序被列为资源根据传输的数据hoggers/接收。该方法使用Taintdroid或Droidbox已修改Android版本，而DroidAnalyst是一款Android平台中立的沙箱。

1. 结论与讨论

Android是一个核心交互平台，提供无处不在的用于连接智能手机的典范服务，因此货币收益，促使恶意软件作者使用各种攻击矢量目标的Android。由于大量增加独特的恶意软件应用程序的签名和内机器人有限能力环境，基于签名的方法是不够的反对看不见，密码和转化代码。研究人员提出了各种行为的方法来防范恶意软件的作者都瞄准方便用户网上发行机制，集中应用程序市场。在本次调查中，我们讨论Android的安全体系结构和它的问题，恶意软件穿透力和隐身技术。在第六节我们讨论静态和动态的方式进行恶意软件分析和检测。这两种方法可以单独使用，但每一个有其自身的局限性。静态分析可通过使用加密和/或转化技术讨论受挫在节V-D。动态分析可以通过多种回避将在第VI-B反模拟技术。我们也包括著名的恶意软件分析和检测方法如表根据他们的目标，方法予和部署。摘要显示没有一个单一的解决方案，解决每一个问题。为了解决各种新恶意软件，进行全面评估框架掺入强大的静态和动态方法可以提出在Android平台。

手动分析已成为不可行由于未知的恶意软件的样本数的指数增长。根据目前的评价，提出了一种自动的，混合方法为Android恶意软件分析。建筑在图中所示的提议的方法。 15是我们今后的研究方向。如图所示APK文件最初解剖静态分析模块。在情况下，其对失败的加密代码，动态分析模块执行行为检测。静态和动态分析将被用来产生应用程序活动报告，以使恶意软件分析确定可疑样本。最后，我们通过突出事实上，混合动力检测方法被越来越突出在恶意软件分析。

