doi:10.3969/j.issn.1000-436x.2014.08.011

Android 恶意软件检测技术分析和应用研究

文伟平，梅瑞，宁戈，汪亮亮
(北京大学 软件与微电子学院，北京 102600)

摘 要：针对 Android 平台安全问题，提出了手机端和服务端协作的恶意代码检测方案，手机端应用主要采用基于 permission 检测技术，实现轻量级的检测。服务端检测系统主要负责对手机端提交的可疑样本进行检测，同时实现了软件行为分析，特征库更新，与手机端同步等功能。其中服务端检测技术包括基于 permission 检测技术、基于字节码静态检测技术和基于 root 权限的动态检测技术。实验测试结果表明，3 种检测技术能达到较好的检测效果。

关键词：恶意代码检测；静态分析；动态分析；权限分析
中图分类号：TP309 **文献标识码**：A **文章编号**：1000-436X(2014)08-0078-08

Malware detection technology analysis and applied research of android platform

WEN Wei-ping, MEI Rui, NING Ge, WANG Liang-liang
(Department of Information Security, School of Software and Microelectronics, Peking University, Beijing 102600, China)

Abstract: For the Android platform security problem, a mobile client and server collaborative malware detection proposal was proposed, where mobile client application was mainly based on permission detection technology and implemented lightweight testing. The server-side detection system is mainly responsible for testing suspicious samples submitted by the mobile terminals, meanwhile implements the functions of software behavior analysis, signature library updates, and mobile client synchronization, etc. The server-side detection techniques include permission-based detection technology, bytecode-based static detection technology and root-based dynamic detection technology. The result of the experiment shows that the three detection techniques can achieve better detection results.

Key words: malcode detection; static analysis; dynamic analysis; permission analysis

1 引言

近几年来，Android 平台已经成为了一个非常流行的手机操作系统平台，并且占据了世界上超过一半的手机操作系统市场份额。随着 Android 智能手机与 Android 平板电脑的普及，基于 Android 的恶意软件也发展迅猛，因此基于 Android 平台的恶意代码检测技术就开始被提出。尽管 Android 智能系统相比苹果的 iOS 系统和诺基亚的 Symbian 系统更年轻，一些 Android 平台上的安全检测技术研究已经发布，如系统访问控制机制[1,2]、恶意软件检测[3]、应用权限分析[4]、内核加固[5]等。大量的 Android 系统安全方面的研究表明，Android 平台需要提出更加稳健的安全检测技术。

对于恶意软件检测技术来说，应用程序的特征是非常重要的。根据文献[6]，恶意软件检测存在 2 种通用技术：静态检测和动态检测。2 种技术各有优缺点[7]，在现实中，大量的方法都是同时包含动态和静态检测技术。静态分析涉及到二进制相关的技术，其中包括反编译、逆向分析、模式匹配和静态系统调用分析等。静态分析技术有一个共同的特点：应用程序不被执行。基于 Android 平台手机端

收稿日期：2013-05-02；**修回日期**：2013-09-17
基金项目：国家自然科学基金资助项目（61170282）
Foundation Item: The National Natural Science Foundation of China(61170282)

应用的扫描技术一般都采用签名静态比对。静态分析的优点是简单并且快速，但是它最大的缺点是扫描恶意软件前需要知道已知恶意软件信息，如签名、行为模式、权限申请等，使得它不能实现自动扫描并适应未知恶意程序的功能。另外一种检测技术是动态检测技术，它的核心过程将应用程序运行在一个封闭的环境并进行监视，从而分析应用程序的行为特征。有很多的参数都可以被动态分析采集，如文件权限改变、进程和线程运行情况、系统调用情况、网络访问情况等。因为动态检测需要应用程序实时运行，并且需要较长的时间采集应用程序的动态数据，所以它比静态分析更复杂。

作者提出了一种基于 permission 组合的技术，帮助用户及时发现潜在恶意应用程序。同时，提出了基于云端静态分析和动态分析技术的方案，为恶意程序的分析奠定坚实的基础。

2　检测框架概述

本文提出的 Android 平台恶意代码检测系统框架由手机客户端和远程服务端组成，手机客户端是一款基于 Android 平台的应用程序，远程服务端收集了大量的资源和数据，以最快的速度检测 Android 应用程序。首先，手机客户端应用程序被发布到 Google 市场或第三方市场，用户可以从市场下载并安装。此应用程序主要负责轻量级的静态扫描和数据的收集工作，手机扫描确定性结果将会直接展示给用户，不能确定的结果将会作为样本上传到远程服务器，供进一步的分析。这里扫描的对象是基于 Andorid 平台的第三方应用程序，它们可以从 Google 市场或者第三方市场下载获得，其中不包括 Android 系统自带程序和厂商自带程序。协作框架如图 1 所示。

图 1　手机端和服务端协作框架

远程服务端检测系统将手机客户端收集的样本数据首先与恶意软件数据库进行比对，如存在相同记录，则直接反馈给手机端，否则启动服务端检测分析系统。服务端检测分析系统包括静态分析和动态调试 2 个模块，由于服务端系统具有丰富的软硬件资源和庞大的特征数据库，会及时针对待分析的样本进行响应。测试若发现未知的恶意软件，首先将该恶意软件的样本特征保存到恶意软件数据库中，然后反馈结果到手机端。服务端检测系统中的静态检测模块主要包括恶意软件静态行为模式分析和基于权限的检测，模式分析检测技术是基于字节码的；动态检测是在一个安全的沙盒中进行的。这个沙盒[8]可以定义为一个执行环境，此环境中所有的进程的行为都要经过安全策略的检查。现实中，沙盒系统中的一个实例，它被系统隔离，不能进行任何恶意行为。

远程服务端检测系统的检测数据来源于 Google 标准市场以及国内市场，还有用户手机端收集到的样本数据。检测系统包括静态检测和动态检测。其中静态检测以缩小检测范围，发现具有疑似恶意行为特征的软件为目标；动态检测则通过动态行为跟踪从而发现未知恶意软件。检测框架如图 2 所示。

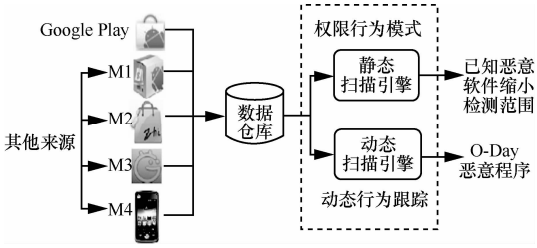


图 2　服务端检测系统框架

3　检测技术

3.1　基于 permission 检测

3.1.1　方法步骤

对基于 permission 检测技术的研究采取归纳，演绎推理的科学实验方法。首先针对实际的恶意程序样本进行分析，提取出它们对权限申请的共同特点，并根据恶意程序样本的分类推理出初步的敏感 permission 组合分类；然后根据初步的敏感 permission 组合分类的规则来匹配海量的应用程序，这些应用程序大都来自 android.market.com, 并将匹配的结果统计出来，结合应用程序的分类产生具体组合规则，这里的分类是 Google 市场上的标准分类；最后使用具体组合规则对新产生的恶意程序进行匹配，观察匹配结果，根据结果再次修正组合规则。流程图如图 3 所示。

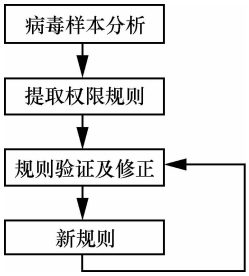


图 3 基于 permission 检测流程

3.1.2 检测策略

3.1.2.1 样本分析

恶意程序样本是 permission 检测技术的实验依据。目前的恶意程序样本库中包括木马远程控制、窃取用户隐私、发送扣费短信和 root 设备 4 大类。每个大类中又有更细的分类，其中木马远程控制包括 Pjapps、Bgserv、Geinimi 和 Rootcager 等几种恶意程序；窃取用户隐私包括 Adrd、Walkinwat、Ewallsh 和 Tapsnake 等几种恶意程序；发送扣费短信包括 FakePlayerA、FakePlayerB、Smstibook、SmsReplicator 和 Adsms 等几种恶意程序；root 设备包括 Zeahache 恶意程序。每种恶意程序都会有特定的攻击步骤，并申请了与其攻击操作相对应的权限。

恶意程序样本的权限提取可以通过 apktool 工具和 shell 脚本完成。首先将所有的恶意程序文件放入一个恶意程序文件夹中，通过 shell 脚本完成遍历所有文件并进行“apktool d”命令，从而获取有效的 AndroidManifest.xml 文件；然后通过 shell 脚本将所有恶意程序的 AndroidManifest.xml 文件中权限记录导入到文本文件中；最后将文本文件转化成 excel 格式并保存。

表 1 中列举了几种典型的恶意软件[9]的申请重要权限列表和攻击行为描述。

表 1 中列举出的 10 种恶意软件家族大都是 2012 年上半年报道出来的。在这些恶意软件中，Geinimi、ADRD、Pjaaps 和 Bgserv 类型的恶意程序一般都具木马远程控制的能力。这些类型的恶意程序都会网络访问功能，有些具有短信监听或发送功能。

3.1.2.2 规则验证

1) 单个危险权限

应用软件中如果申请了 Android 系统中的危险权限，此应用具有较高的危险性。从统计学的角度分析，此类应用程序一般具有特殊的功能，而且应用市场的应用绝大多数都不会申请此类敏感权限，所以首先将具有此类权限申请的应用筛选出来。单个危险权限规则如表 2 所示。

以上权限都是敏感权限，都被标识为：dangerous，signature 或者 signatureOrSystem。例如，在申请 SET_DEBUG_APP 权限后，程序就具有配置一个程序用于调试的能力。Android SDK 中有些 API 对开发者透明，这些 API 不能被第三方应用程序调用，仅能被系统程序调用。恶意程序的作者可以下载包含隐藏 API 的 Android 源代码，编译，修改并且生成应用程序。因此，恶意程序通过申请

表 1 恶意软件权限申请及攻击行为描述

恶意软件	重要敏感权限	描述
ADRD	INTERNET, ACCESS_NETWORK_STATE, RECEIVE_BOOT_COMPLETED	蠕虫病毒
DroidDream	CHANGE_WIFI_STATE	root 提权利用木马
Bgserv	INTERNET,RECEIVE_SMS,SEND_SMS	蠕虫病毒
DroidDreamLight	INTERNET,READ_PHONE_STATE	信息窃取木马
Genimi	INTERNET,SEND_SMS	蠕虫病毒
jSMSHider	INSTALL_PACKAGES	破坏系统固件木马
Pjapps	INTERNET,RECEIVE_SMS	蠕虫病毒
Zsone	RECEIVE_SMS,SEND_SMS	恶意发送短信木马
zHash	CHANGE_WIFI_STATE	root 提权利用木马
BaseBridge	NATIVE_CODE	root 提权利用木马

表 2　　单个危险权限规则

权限	能力描述
SET_DEBUG_APP	为调试配置一个应用程序
SET_TIME	允许应用程序设置系统时间
SET_TIME_ZONE	允许应用程序设置系统时区
WRITE_APN_SETTINGS	允许应用程序设置 APN，其中的 APN（acess point name）即“接入点名称”，用来标识 GPRS 的业务种类，目前分为 2 大类：CMWAP(通过 GPRS 访问 WAP 业务)、CMNET（除了 WAP 以外的服务目前都是 CMNET,比如连接因特网等）
BRICK	禁用设备必须拥有的权限
INSTALL PACKAGES	安装新的软件
REBOOT	重启设备
SHUTDOWN	关闭设备
CHANGE_WIFI_STATE	改变 Wi-Fi 的状态

表 3　　危险权限组合规则

权限	描述
INTERNET,ACCESS_NETWORK_STATE, RECEIVE_BOOT_COMPLETED	木马控制能力
READ_CONTACTS , INTERNET	泄漏用户联系人的能力
INTERNET,RECEIVE_SMS,SEND_SMS	在后台实现扣费短信能力
INTERNET,READ_PHONE_STATE	泄漏 IMEI 号能力
INTERNET,SEND_SMS	木马控制能力
INTERNET,RECEIVE_SMS	木马控制能力
RECEIVE_SMS,SEND_SMS	后台实现扣费短信能力
RECEIVE_BOOT_COMPLETED & INTERNET & ACCESS_FINE_LOCATION	跟踪用户地理位置的能力，跟踪方式为 GPS
RECEIVE_BOOT_COMPLETED & INTERNET & ACCESS_COARSE_LOCATION	跟踪用户地理位置的能力，跟踪方式为 CELL-ID 或者 WIFI
PROCESS_OUTGOING_CALL & RECORD_AUDIO & INTERNET	具有记录用户打电话的内容并且上传到指定服务器能力
READ_CONTACTS & SEND_SMS	泄漏用户联系人并发送短信能力

SET_DEBUG_APP 权限，有可能禁止一些反恶意程序软件的运行。

2) 多个危险权限组合

申请的权限越多，恶意程序发挥的空间越大，对用户威胁越大。很多恶意程序在攻击之前，向系统申请大量的权限，以便成功调用具有威胁的 API。通过对上述恶意程序样本攻击特点的分析，列举出多个危险权限组合的规则，如表 3 所示。

3.2　字节码静态检测

3.2.1　*方法步骤*

本文的静态分析技术是基于 ASM 字节码处理框架的字节码解析技术。它的目标是跟踪恶意程序的静态行为并标识。基本步骤是：首先解压 APK 分组，并通过 dex2jar 工具将 classes.dex 文件转化为 jar 分组，然后分析字节码指令，通过 ASM 框架获取到函数调用关系和 JVM 运行指令。在分析过程中，模拟 JVM 指令运行，得到函数的静态参数值和函数之间的调用关系图，通过静态分析函数之间的调用关系以及相关度来确定恶意软件的行为。字节码的分析由 shell 脚本、Java 分析以及一些 python 脚本组成。具体的操作步骤如图 4 所示。

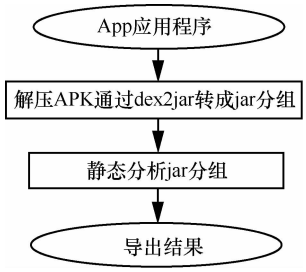


图 4 字节码检测流程

基于字节码的静态分析前期准备需要获取到 jar 分组，其中包含了应用程序的 class 文件。Android 应用程序中都包含 AndroidManifest.xml 文件，此文件中包含了应用程序的权限申请和 Activity、Broadcast 以及 Provider 的声明。字节码分析的入口以 Activity 为基础；对于 Broadcast 声明，一般都由 onreceive 函数来接受广播消息，因此 onreceive 也是一个分析入口。具体的工作是分析 jar 分组中的 class 文件，分析内容包括查找敏感函数、跟踪静态参数和恶意软件行为模式分析。

3.2.1.1 获取 jar 分组

首先将 APK 文件后缀改为 zip 并解压，得到其中的 class.dex 文件，它就是 Java 文件编译再通过 dx 工具打包而成的。下载 dex2jar 工具，将 class.dex 复制到 dex2jar.sh 所在目录，并在其目录下运行 sh dex2jar.sh class.dex 命令，生成一个文件 classes_dex2jar.jar。此时的 jar 分组就是静态检测需要的并分组文件。

3.2.1.2 查找敏感函数

应用程序申请特定权限后，是否调用了敏感函数有待于静态代码的分析。敏感函数跟踪技术利用 ASM 基本框架，遍历软件中导入的系统库函数，查找敏感函数，从而确定是否调用了敏感函数。从 ASM 框架提供的 API 中，可以获取到所有的类及其函数的详细信息，包括类名称、函数名称、参数等。注意用户定义的类或者函数可能被混淆，但系统函数是不能混淆的，而跟踪的敏感函数都为系统函数，所以混淆技术并不影响基于字节码的跟踪。

3.2.1.3 跟踪静态参数

由 3.2.1.2 节可知，恶意软件通过调用敏感函数，实现了恶意行为。为了进一步跟踪恶意软件的行为，作者提出了一种跟踪敏感函数中静态参数的方法。一条字节码指令由操作指令和固定参数组成。JVM 操作堆栈的基本原则是：操作符对应这一个或者多个操作数，操作数如何执行由操作符安排。

3.2.2 行为模式分析

根据上文介绍的恶意软件家族的行为模式，作者总结出具有恶意行为的基本路径模式。恶意软件行为模式分析可分为 2 个步骤：一是构建应用程序的全部路径图，路径的入口从 AndroidManifest.xml 文件中获取，主要是 Activity 声明的类；二是将具有恶意行为的路径集合与应用程序全部路径图进行比对，从而决定应用程序是否具有恶意行为。

全部路径图的构建是基于字节码的分析结果。Class 文件中都包含具体的方法和变量，在这里，为了简化处理提高检测效率，忽略成员变量的分析。Class 文件中的方法包括系统 API 以及用户自定义的 API，为了跟踪具体的行为，路径图需要精确到系统的 API，系统的 API 包括类名、函数名以及参数。如 android/telephony/TelephonyManager, getDeviceId, ()Ljava/lang/String。其中 getDeviceId 是获取用户的 IMEI 号的函数，具有泄露用户隐私的功能。恶意软件的行为模式提取来自已知恶意程序的分析将取 3.3 节中介绍。以下算法说明了如何检测恶意软件行为。

```
Input:  entry points, known malware pattern
Output:  true or false
Foreach entry point ∈ entry points do
  methodlist = initial state: start of entry point
    state = initial state
      foreach method ∈ methodlist do
        foreach methodnode ∈ method do
          if methodnode is sysAPI then
          add(methodlist, methodnode)
           else
          track sysAPI and add in methodlist
          end
        end
      end
  end
  if known malware pattern match methodlist then
    return true
  else
    return false
```

在实验环境中，相比较仅仅使用 permission 检测技术，融合此算法后的检测方法可以有效地提高

恶意软件的发现率，同时，通过机器学习和系统函数调用风险库的逐渐积累，还可进一步提高恶意软件的检测率。

3.3　动态检测

3.3.1　方法步骤

动态分析是一个长期且复杂的过程，因为它需要许多的准备工作，而且运行起来效率不高，一般都是通过沙盒进行检测。动态分析的框架如图 5 所示。

作者对 Android 市场上 50 580 个应用程序进行了分析，经过静态检测后，过滤到了 1 300 多个应用程序进行动态分析。为了实现动态分析 APK 的需求，需要准备大量的 APK 程序、root 后的设备、monkeyrunner 相关的 python 脚本、tcpdump 工具、wireshark 数据分组分析工具以及 trace 工具。

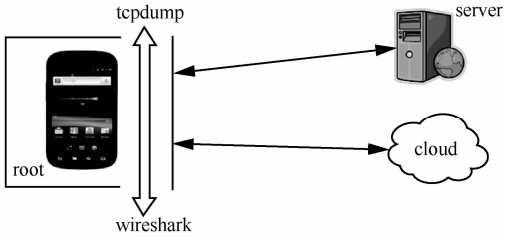


图 5　动态分析框架

本文的动态分析通过 trace 工具记录系统调用的行为并将所用行为写入到日志文件中，同时跟踪分析 wireshark 网络数据分组发送情况，最后将日志文件和数据分组文件统一起来进行分析。基本步骤如下。

1) 准备工作。启动模拟器，它运行在 PC 机的桌面上。当一个应用运行在模拟器上，它可以通过服务调用其他应用，访问网络，打开视频或音频，存储或者遍历文件等。模拟器包含大量的调试功能，如模拟打电话、发短信、adb 日志等。

2) 安装 trace 和 wireshark 工具。动态检测的目标是监视应用运行时的动态行为，这些行为包括调用系统的 API，发送网络数据分组等。通过 adb 工具将 trace 安装上，同时 wireshark 安装在 PC 机端。这里，通过 trace 工具产生的系统调用都被存入日志文件中。

3) 安装应用并启动 mokeyrunner 工具。通过 adb 安装应用程序，一旦应用程序安装成功后，并自我运行。此时，mokeyrunner 工具可以模拟用户行为，点击应用程序，使其运行起来。

4) 收集日志记录和网络数据记录。当应用程序通过 monkeyrunner 点击完成后，日志数据将会存储在文件中，如果有网络交互，将会有网络数据被记录在 wireshark 中。通过对这些数据的分析以及恶意软件的模式匹配可以确定应用程序是否为恶意软件。

3.3.2　检测 Zero-Day 恶意程序

下面将会介绍动态分析技术发现的 Zero-Day 恶意程序。它们是 Plankton 和 DoridKungFu 恶意程序。经过 permission 和字节码静态分析后，作者收集了 1 300 多个可疑的应用程序。本节将会介绍分析发现 Zero-Day 恶意程序的详细过程。

根据是否调用了动态加载对象，进一步经过静态扫描 1 300 个可疑应用程序后，作者发现了 1 024 个应用程序调用了动态加载对象 DexClassLoader 并且加载了其他的应用程序。作者经过进一步的分析，发现 1 024 个应用程序中，大量的应用都使用了广告库，而有些广告库中包含了动态加载的代码。为了排除广告的影响，作者去除包含具有动态加载能力的广告的应用程序，最终得到了 204 个可疑应用程序。下一步对这 204 个应用程序进行动态检测。

根据动态检测的记录，作者不仅发现了动态加载的行为，而且收集到了其动态加载的文件（jar/apk）。在这些可疑应用中，作者发现了一个特例。它的应用名称是愤怒的小鸟官方版本，包名是经过愤怒的小鸟的分组名改装而来，具体是 com.crazyapps.angry.birds.cheater.trainer.helper。从动态检测的日志中看出，它动态加载了 jar 分组：plankton v0.0.4.jar，而这个 jar 分组下载的地址来自一个远程服务器。更进一步的分析，发现它在下载 jar 之前，收集应用程序的 permission，并把手机的结果上传到远程服务器。这样做，估计是服务器要根据客户端的 permission 来决定动态加载的 jar 分组。通过对 plankton v0.0.4.jar 的分析，作者发现其中包含的构建僵尸网络的功能，并且具有监听远程服务端命令的特点。这些监听命令包括 BOOKMARKS, INSTALLATION, STATUSD 等。具体命令如下。

```
public enum Commands
{...
static
{
ACTIVATION = new Commands ("ACTIVATION", 1, "Activation", "/activate");
```

```
HOMEPAGE = new Commands("HOMEPAGE", 2, "Homepage", "/homepage");
COMMANDS_STATUS = new Commands ("COMMANDS_STATUS", 3, "CommandsStatus", "/commandstatus");
BOOKMARKS = new Commands ("BOOKMARKS", 4, "Bookmarks", "/bookmarks");
SHORTCUTS = new Commands ("SHORTCUTS", 5, "Shortcuts", "/shortcuts");
HISTORY = new Commands("HISTORY", 6, "History", "/history");
TERMINATE = new Commands ("TERMINATE", 7, "Terminate", "/terminate");
STATUS = new Commands("STATUS", 8, "Status", "/status");
DUMP_LOG = new Commands("DUMP_LOG", 9, "DumpLog", "/dumplog");
UNEXP_EXCEPTION = new Commands ("UNEXP_EXCEPTION", 10, "UnexpectedException", "/unexpectedexception");
UPGRADE = new Commands("UPGRADE", 11, "Upgrade", "/installation");
INSTALLATION = new Commands ("INSTALLATION", 12, "Installation", "/installation");
arrayOfCommands[0] = COMMANDS;
arrayOfCommands[1] = ACTIVATION;
arrayOfCommands[2] = HOMEPAGE;
arrayOfCommands[3] = COMMANDS_STATUS;
arrayOfCommands[4] = BOOKMARKS;
arrayOfCommands[5] = SHORTCUTS;
arrayOfCommands[6] = HISTORY;
arrayOfCommands[7] = TERMINATE;
arrayOfCommands[8] = STATUS;
arrayOfCommands[9] = DUMP_LOG;
arrayOfCommands[10]= UNEXPECTED_ EXCEPTION;
arrayOfCommands[11] = UPGRADE;
arrayOfCommands[12] = INSTALLATION;
}
}
```

发现 Plankton 恶意程序后，作者提取出它的签名并将其报告给某反病毒厂商内部并扫描 50 580 个应用程序，发现了 10 款同样签名的应用。这些应用中都包含 Plankton 恶意程序的攻击特点。

经过静态扫描后，作者收集到了 2 180 个具有访问 native-code 的应用。一般的情况下，native-code 被存入 libs/lib 的目录下面，有些带有恶意行为的应用会将其存在其他的目录中，如 raw、assets 目录。根据目录规则静态扫描 2 180 个应用，作者发现在 408 个应用中，native-code 被存入不正常的目录中。对 408 个可疑应用动态分析后，根据记录的日志，作者发现一些应用多次尝试调用 sys_mount 系统函数并修改系统文件。这些应用具有极高的可疑性，因为这些系统调用的函数只有 root 权限才能满足。对于第三方应用来说，它极有可能已经获取了 root 权限。根据这个思路，作者发现了具有 root 漏洞利用的恶意程序：DroidKungFu。这个恶意程序包含了 2 种 root 漏洞利用的代码：Rageagainstthecage 和 Exploid，并且对代码进行了加密。当恶意程序启动时，它会去判断是否有 root 权限，如果有，就修改系统文件并安装具有 root 权限的应用，否则启动 root 漏洞利用程序，获取 root 权限。

4 实验数据分析

本文提出的检测方案包括手机端和云端协作检测和服务端的检测系统。手机端轻量级检测包括静态扫描和上传可疑应用。通过发布手机端的应用程序，作者收集 15 895 个应用。服务端检测系统的数据大都来自市场。通过网络爬虫的收集，作者获取了 34 685 个应用。这些应用有些来自不同的市场，所以需要排除其中签名相同的应用。经过同签名应用的合并操作，最终的分析数据为 45 820。

服务端检测是本文的重点内容，它包括静态检测和动态检测。静态检测运行速度快，可以迅速的缩小动态检测的范围。通过对 50 580 个应用进行分析，静态检测的结果如表 4 所示。

表 4 静态分析后统计结果

其他数据来源	恶意倾向软件	比例/%	恶意软件
Google 市场（20 925）	130	0.62	3
M1（5 225）	252	4.82	8
M2（3 665）	187	5.10	7
M3（4 870）	233	4.78	6
M4（15 895）	577	3.63	23
总数（50 580）	1 379	2.73	47
不同应用（45 820）	1 184	2.58	45

从表中数据可以看出，静态扫描分析后，可以确定 47 个已知恶意软件，其中包括 45 个不同的恶意软件。由于静态检测的准确度有限，不能完全确定恶意程序，所以过滤后的软件为具有恶意倾向的软件。在 50 580 个应用程序中，静态分析后，发现 1 379（2.73%）个应用存在恶意倾向。从分析的比例中可以看出，静态分析具有较好的分析效果，因为它大大地缩小了检测的范围，并且检测速度相对较快。然而对具有恶意倾向应用程序的好坏的完全鉴定，需要动态分析或者人工分析的进一步工作。

从静态分析的结果中，导出具有可疑倾向的应用程序，将其放入动态分析系统中。本文对 1 379 个应用程序进行了动态分析，最终的结果如表 5 所示。

表 5　　动态分析后统计结果

其他数据来源	恶意软件	比例%
Google 市场（20 925）	10	0.05
M1（5 225）	20	0.39
M2（3 665）	15	0.41
M3（4 870）	22	0.45
M4（15 895）	51	0.32
总数（50 580）	118	0.23
不同应用（45 820）	108	0.23

动态分析针对应用程序的动态行为进行分析，包括动态调用 API 的行为、网络数据分组发送行为、硬件设备调用行为以及联系人等隐私数据的操作行为。动态分析之前需要考虑第三方广告申请的权限。例如，静态分析从 Google 市场下载的测试数据，得到有 130 个恶意倾向程序需要进一步分析，但是作者发现其中有不少应用都加载了广告并申请了特殊权限，所以过滤掉这些带有广告的应用后，只剩下 43 个应用程序需要进一步进行动态分析，从而减轻了动态分析的负担。

从最终的分析结果可知，服务端检测系统检测了 50 580 个应用程序，发现了 118 个恶意程序。检测比例为 0.23%。检测恶意签名数目为 108。通过服务端检测系统，最终，作者发现了 2 款 Zero-Day 恶意程序，分别为 Plankton 和 DroidKungFu。总之，从 Zero-Day 恶意程序的具体分析过程可以看出，一般情况下，Zero-Day 恶意程序的发现需要人工分析。

5　结束语

本文提出了恶意软件检测的 3 项基本技术：permission 分析、字节码分析和动态分析。Permission 分析有效地缩小了检测的范围，字节码检测技术主要是通过 ASM 框架进行字节码分析，从而实现关键函数查找和恶意程序行为分析的技术。这种基于字节码的检测技术不能完全解决恶意程序分析工作，因为它存在这样的问题：如果一个恶意程序将静态信息存储在代码中，基本上能检测出来，但如果实现动态加载数据，此方法无效。

基于静态检测技术的不足，本文提出了实时监控应用程序行为的动态检测技术。但由于 Android 平台本身安全机制的限制，动态监控需在 root 权限下工作。动态检测技术解决了静态检测技术不能解决的问题。2 种技术结合起来，为 Android 平台上的恶意程序检测工作提供了有力的保障。Android 平台上的动态检测有其天生的弱点：分析时间长，工作量大。大量的实验证明，这种技术能准确地检测出恶意软件，可以大大减轻后期人工分析恶意程序特征的工作。

从分析的准确度来看，3 种技术的准确度逐步增加，从技术难度来看，3 种技术难度逐步增加。3 种技术是前后关联的有机整体，基于 permission 检测技术主要的目的是从海量的数据中分析出具有可疑的恶意软件，基于字节码的静态检测技术进一步从可疑的恶意软件中去分析具体的静态行为，基于 root 权限的动态检测技术结合静态分析技术来确定恶意软件以及其攻击特征。

参考文献：

[1] JESSE B. Developing secure mobile application for Android[EB/OL] https://www.isecpartners.com/files/iSEC_Securing_Android_Apps.pdf, 2008.

[2] SCHMIDT A D, SCHMIDT H G, BATYUK L. Smartphone malware evolution revisited: Android next target[A]. Proceedings of the 4th IEEE International Conference on Malicious and Unwanted Software [C]. USA, 2009. 1-7.

[3] SCHMIDT A D, SCHMIDT H G, CLAUSEN J. Static analysis of executables for collaborative malware detection on android[A]. IEEE International Congress on Communication (ICC) 2009 - Communication and Information Systems Security Symposium[C]. 2009.

[4] ENCK W, ONGTANG M, MCDANIEL P. Understanding Android security[J]. IEEE Security and Privacy, 2009, 7(1):50-57.

[5] SHABTAI A, FLEDEL Y, ELOVICI Y. Securing android-powered mobile devices using selinux[A]. IEEE Security and Privacy[C]. 2009.10-15.

[6] BERGERON J, DEBBABI M, DESHARNAIS J. Static detection of malicious code in executable programs[A]. Proceedings of the Symposium on Requirements Engineering for Information Security[C]. USA, 2001.20-24.

（下转第 94 页）

1459-1472.

[2] 尹震宇, 赵海.WSN 中基于分簇路由的多维度数据压缩算法研究[J]. 电子学报, 200937(5): 1109-1114.
YIN Z Y, ZHAO H. Research on multi-dimensional data compression algorithm for cluster-based routing in wireless sensor network[J]. Acta Electronica Sinica, 2009, 37(5):1109-1114.

[3] GANESAN D, GREENSTEIN B, ESTRIN D, *et al*. Multiresolution storage and search in sensor networks[J]. ACM Transactions on Storage, 2005, 1(3):277-315.

[4] SHEN G, ORTEGA A. Joint routing and 2D transform optimization for irregular sensor network grids using wavelet lifting[A]. Proc of IPSN[C]. St Louis, USA, 2008.

[5] NARANG S K, SHEN G, ORTEGA A. Unidirectional graph-based wavelet transforms for efficient data gathering in sensor networks[A]. Proc of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)[C]. Dallas, USA, 2010.

[6] LIU K.H, TENG W G, CHEN M S. Dynamic wavelet synopses management over sliding windows in sensor networks[J]. IEEE Transactions on Knowledge and Data Engineering, 2010. 22(2):193-206.

[7] REIN S, REISSLEIN M. Low-memory wavelet transforms for wireless sensor networks: a tutorial[J]. IEEE Communications Surveys & Tutorials, 2011, 13(2):291-307.

[8] 周四望, 林亚平. 无线传感器网络中的小波方法[M]. 湖南: 湖南大学出版社, 2011.
ZHOU S W, LIN Y P. Wavelet Methods for Wireless Sensor Network[M]. Hunan: Hunan University Press, 2011.

[9] 胡玉鹏, 林亚平, 周四望等. 面向异步通信机制的无线传感器网络及其 MAC 协议研究[J]. 计算机学报, 34(8): 1463-1477, 2011.
HU Y P, LIN Y P, ZHOU S, *et al*. Asynchronous communication mechanism oriented wireless sensor networks and MAC protocols[J]. Chinese Journal of Computers, 2011,34(8):1463-1477.

[10] YANIV R, BURSHTEIN D. An enhanced dynamic time warping model for improved estimation of DTW parameters[J]. IEEE Transactions on Speech and Audio Processing, 2003, 11(3):216-228.

[11] HEINZELMAN W, CHANDRAKASAN A, BALAKRISHNAN H. An application-specific protocol architecture for wireless microsensor networks[J]. IEEE Transactions on Wireless Communications, 2002, 1(4):660-670.

作者简介：

周四望（1971-），男，湖南岳阳人，博士，湖南大学副教授，主要研究方向为传感器网络信息处理、压缩感知与小波分析。

李兰（1984-），女，福建福州人，湖南大学硕士生，主要研究方向为传感器网络信息处理。

（上接第 85 页）

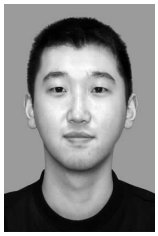
[7] MOSER A, KRUEGEL C, KIRDA E. Limits of static analysis for malware detection[A]. Proceedings of the 23rd Annual Computer Security Application Conference[C]. Seoul, Korea, 2007.421-430.

[8] BISHOP M A. The Art and Science of Computer Security[M]. Boston: Addison-Wesley Longman Publishing Co, 2002.213-217.

[9] http://www.symantec.com/security_response/writeup.jspdocid=2011-022303-3344-99[EB/OL].2001.

作者简介：

文伟平（1976-），男，湖南益阳人，博士，北京大学副教授，主要研究方向为网络攻击与防范、软件安全漏洞分析、恶意代码研究、信息系统逆向工程和可信计算技术等。



梅瑞（1984-），男，安徽六安人，北京大学硕士生，主要研究方向为网络与软件安全、信息系统风险评估等。

宁戈（1988-），男，山西大同人，北京大学硕士生，主要研究方向为系统及网络安全、漏洞分析及利用技术等。

汪亮亮（1987-），男，安徽安庆人，北京大学硕士生，主要研究方向为系统及网络安全、网络攻击与防范等。