# 基于AOP的Android Studio恶意

# 代码检测

# 郑文浩/1 白璠/1 丹增晋美/2

北京理工大学

501320528@qq.com

# **abstract**：近年来随着智能手机的发展，基于Android Studio的恶意代码逐渐增多，防止和检测恶意代码就显得尤为重要，本文提出了一种通过AOP获取代码的CPU占用率与内存使用、APP权限等情况，再通过机器学习的算法，用训练集来生成决策树模型，通过切面获取用例信息组成测试集来检测恶意代码。

# **导言**

**1.1背景**

# 随着移动智能终端设备快速发展升级，手机已经远远超越了打电话，发短信的基本功能；移动智能终端的计算能力、存储能力和智能化程度已经可以和普通的个人 PC设备相当，这就给恶意代码应用的运行提供了运行的硬件环境。再加上目前4G网络的大力推广，移动互联网又进入飞速发展的阶段。移动电视应用伴随4G的到来都会获得进一步推广发展。中国互联网协会发布公告称，2013年中国移动互联网已达到了 310亿元的规模。移动平台会逐渐成为人们上网的主要方式。当用户享受着快速网络提供的便利的同时，也同时不得不面对严峻的移动安全问题。近日，著名安全厂商 F-Secure发布了一份安全报告。报告称， 2013年Android平台滋生的恶意应用占所有手机应用的 97%，而这一数据在 2012年为 79%。根据 F-Secure的说法，安卓平台恶意应用数目从 2012年的 238种上升到804个

# 新类型和变体。

# 2013年全年用户举报的各种类型骚扰电话号码数量超过4.3亿，用户接收到的垃圾短信数目，也超过了 4.5亿条。与桌面端的恶意应用程序的扩散方式不同，移动端的恶意应用大部分经由非谷歌官方的应用商店、二维码、论坛等方式进行扩散。由于应用商店、论坛的管理不严格，对开发者提供的移动应用程序审核不过关，同时缺乏行之有效的恶意代码检测手段，使得现今移动网络上恶意代码问题越来越严重。

# 截止到现在 Android平台上的恶意移动应用APP的恶意行为主要包括：恶意消耗用户资费、盗取用户敏感信息、远程控制 Android智能平台、恶意的占用移动终端的软硬件资源，还有利用 Android移动智能终端传播恶意移动应用 APP等，恶意移动应用 APP比较常见的恶意行为表现为恶意用户资费消耗和盗取用户敏感数据。

# (1)恶意消耗用户资费

# 恶意消耗用户资费是最经常出现的恶意移动应用 APP，主要是通过订购增值业务和利用手机进行支付消费；这些的恶意行为都直接给用户造成了经济上的损失。

# (2)盗取用户的敏感信息

# 这也是用户经常遇到的恶意移动应用类型，主要的体现在获取用户的通信数据（包括：通话记录、短信或彩信内容、邮件的内容、地理位置信息和通信录联系人数据）和移动应用 APP中用户隐私信息（包括：用户的账号、密码信息和应用隐私数据）。如果用户的 Android移动智能终端安装了这类的恶意移动应用APP那么移动智能终端中包含的通信数据和移动应用 APP中用户的隐私信息都存在隐私信息泄露的威胁。

# (3)远程控制

# 这类恶意移动应用 APP主要在后台悄悄的接收远程终端的控制命令，并将命令执行的结果发送到控制端，使用户的移动智能终端变成一个傀儡机。

# (4)恶意占用移动智能终端的软硬件资源

# 恶意移动应用 APP会在启动后台服务，在后台运行一些操作，强行的占用了系统的内存空间和一定的 CPU资源。拖慢了移动智能终端的运行速度，降低了移动智能终端的用户体验。

# (5)利用智能终端传播恶意应用

# 现在的移动智能终端都具有多种的数据传输交换的能力，该类病毒利用了移动智能终端提供的通信信道以短信、彩信、邮件和蓝牙等无线通信手段传播恶意应用的代码进行自我复制。这些恶意移动应用 APP带来的种种恶意行为严重侵害了广大用户的合法权益，甚至在某种程度上影响了公众对于智能终端安全性的信心，给移动智能终端产业的进一步发展带形成阻碍。

# **1.2国内外研究现状**

# 目前，移动安全威胁引发全球关注。国内外的安全技术研究人员对无线平台安全分析技术与检测技术做了大量的研究工作。

# 随着移动智能终端的不断的发展，它的运算能力已经赶上甚至超越了早期的PC机。移动智能终端和个人电脑之间差距越来越小，与私人电脑上的软件安全检测技术发展的过程相类似，移动智能终端上的应用 APP安全检测技术也是先从静态检测部分开始的。Anthony Desnos，Geoffroy Gueguen提出了Android移动应用APP的静态分析Androguard方法，这个方法可以对Android的字节码进行分析，并基于此给出了一种基于恶意特征签名的恶意软件检测技术。

# Taintdroid方法。该方法可以有效的追踪系统内的敏感数据在系统内的流动，使用户能及时发现移动应用是否泄漏了用户敏感数据。系统监控智能移动终端中的敏感数据，依据程序代码对隐私数据的使用情况来判断其是否为恶意代码。首先被标记为污染源，一个污染标记指明此信息已经被分配好。动态污点分析技术追踪一个标签数据在传递过程中可能通过某种途径泄漏的敏感信息。这种追踪方式往往在指令级水平执行。最终，受影响的数据在离开系统时，提取出其中的污点标签，并记录下应用程序的行为。

# 综上所述，针对移动智能终端的安全检测技术主要分为：基于权限的恶意软件检测技术、基于特征码的检测技术和基于行为的恶意软件检测技术三种检测技术方法。

本文介绍了一些能有效地识别这些恶意软件的方法，将它们组合起来生成决策树综合地对APP进行判定，希望解决面前存在的恶意软件大量充斥在应用市场的问题。

1. **Aspectj介绍**

“a seamless aspect-oriented extension to the Javatm programminglanguage”（一种基于Java平台的面向切面编程的语言）。Aspectj能做什么干净的模块化横切关注点，如错误检查和处理，同步，上下文敏感的行为，性能优化，监控和记录，调试支持，多目标的协议。

AspectJ的动机是发现那些使用传统的编程方法无法很好处理的问题。@AspectJ 使用了Java5 的注解，可以将切面声明为普通的Java类。

[面向方面编程](http://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E6%96%B9%E9%9D%A2%E7%BC%96%E7%A8%8B" \t "http://baike.baidu.com/_blank)的出现正好给处于黑暗中的我们带来了光明，它针对于这些横切关注点进行处理，就好像面向对象编程处理一般的关注点一样。而作为AOP的具体实现之一的AspectJ，它向Java中加入了连接点（Join Point）这个新概念，其实它也只是现存的一个Java概念的名称而已。它向Java语言中加入少许新结构：切点（pointcut）、通知（Advice）、类型间声明（Inter-type declaration）和方面（Aspect）。切点和通知动态地影响程序流程，类型间声明则是静态的影响程序的类等级结构，而方面则是对所有这些新结构的封装。

**3.数据挖掘**

本次研究采用动态分析，用过切面程序基于权限、CPU占用率与内存使用情况分析项目。

通过设置切点，编写切面程序，在项目当中获取所需的数据。

以下为本次实验调取权限、CPU占用率与内存使用情况的切面程序的部分代（主要部分）。其中获取系统各项数据的方法被我们封装在SystemInformation类中。

@Aspect  
public class TraceAspect {  
 private static final String METHOD\_EXECUTION = "execution(\* \*..MainActivity.\*(..))";   
 @Pointcut(METHOD\_EXECUTION)  
 public void methodExecution() {  
 }  
@TargetApi(Build.VERSION\_CODES.LOLLIPOP)  
 @Before("methodExecution()")  
public void beforeMethodExecution(JoinPoint joinPoint) {

//获取方法名

getgetSignature().getName();

//获取开始时间

SystemInformation.getstartTime()  
 }  
@TargetApi(Build.VERSION\_CODES.LOLLIPOP)  
@After("methodExecution()")  
public void SystemInformation.afterMethodExecution(JoinPoint joinPoint) {

//获取结束时间

SystemInformation。GetEndTime();

//获取CPU使用率  
SystemInformation.getProcessCpuRate();

//获取可用物理内存

SystemInformation.getFreePhysicalMemory()

//获取软件获得的系统权限

SystemInformation.getPermission(Context context)   
 }

}

通过以上的AspectJ代码片段，我们可以获取到APP的对手机的各项权限以及手机状态，通过下面的数据分析将数据进行整理得出结论

**4.数据分析**

对提取的数据本次实验采用机器学习的方法，机器学习理论主要是设计和分析一些让计算机可以自动学习的算法。机器学习算法是一类从数据中自动分析获得规律，并利用规律对未知数据进行预测的算法。因为学习算法中涉及了大量的统计学理论，机器学习与统计推断学联系尤为密切，也被称为统计学习理论。

此次采用当中的j48决策树进行分析决策树生成算法，算法决策树是一种通过对历史数据进行测算实现对新数据进行分类和预测的算法。简单来说决策树算法就是通过对已有明确结果的历史数据进行分析，寻找数据中的特征。并以此为依据对新产生的数据结果进行预测。

**(1)决策树生成**

本次借助开源数据挖掘工具weka的jar架包，通过api调用其中的决策树生成算法，并生成图形显示，部分代码如下所示

Classifier m\_classifier = new J48();

File inputFile = new File("train.arff"); // 训练语料文件

ArffLoader atf = new ArffLoader();

atf.setFile(inputFile);

Instances instancesTrain = atf.getDataSet(); // 读入训练文件

instancesTrain.setClassIndex(0);m\_classifier.buildClassifier(instancesTrain); // 训练

inputFile = new File("test.arff"); // 测试语料文件

atf.setFile(inputFile);

Instances instancesTest = atf.getDataSet(); // 读入测试文件

// 设置分类属性所在行号（第一行为0号，instancesTest.numAttributes()可以取得属性总数

instancesTest.setClassIndex(0);

double sum = instancesTest.numInstances(); // 测试语料实例数

double right = 0.0f;

// 测试分类结果

for (int i = 0; i < sum; i++) {

// 如果预测值和答案值相等（测试语料中的分类列提供的须为正确答案，结果才有意义）

if (m\_classifier.classifyInstance(instancesTest.instance(i)) == instancesTest.instance(i).classValue()) {

// 正确值加1

right++;

}

}

算法中用到的主要[数据结构](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "http://blog.csdn.net/roger__wong/article/details/_blank)

（1）Instances对象

一个Instances代表一张表，可以对应一个arff文件或者是一个csv文件，通过Instances对象可以取某一列的均值方差等，主要就是若干行记录的一个封装。

（2）Instance

一个Instance代表一行记录，换言之一个Instances的数据包含多个Instance。每个Instance会有一个特殊的ClassIndex，该列值代表该Instance属于哪一类，具体来说就是图一里面的Golf。

（3）Classifier接口

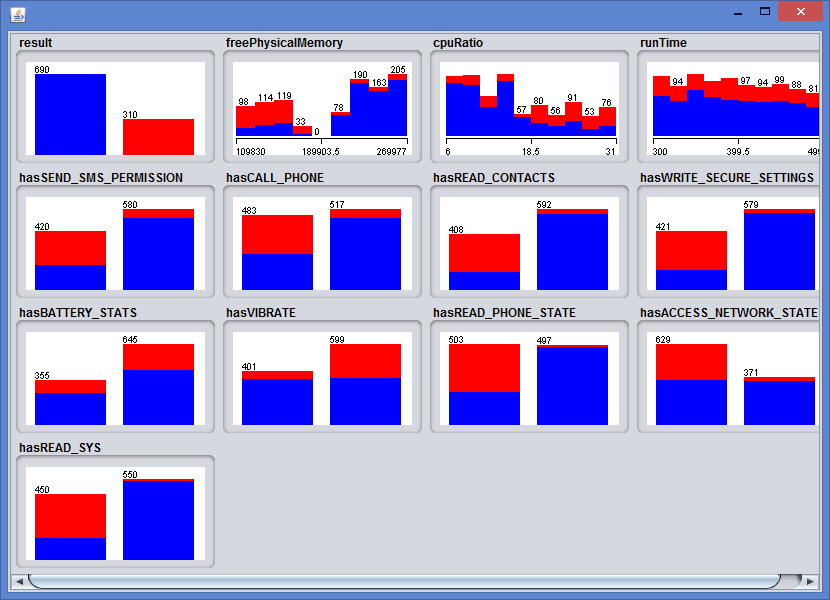
Weka中每一个分类器都继承与这个接口（虽然从意义上来说是个接口但其实是个子类），该接口提供一个buildClassifier方法传入一个Instances对象用于训练，还有classifyInstance方法用于传入一Instance来判断其属于哪个类。

（4）J48

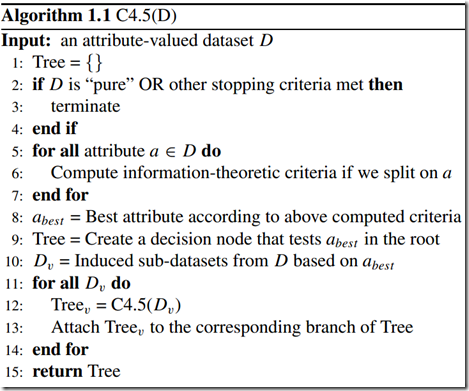
分类器主类，实现了Classifier接口。

（5）ClassifierTree接口代表树中的一个节点，维护和组成树的结构。其中J48用到C45PruneableClassifierTree和PruneableClassifierTree。

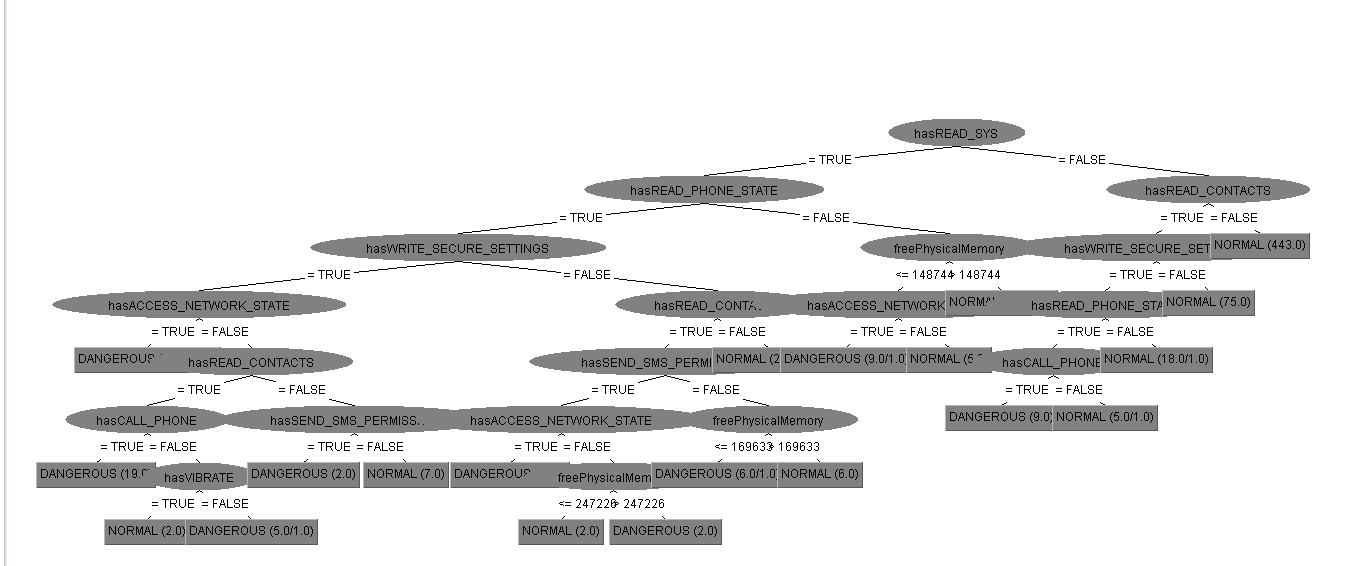
本次数据分析中，训练集由1000个样本组成，我们到网上查找了相关的论文并对他们的分析数据进行整理，构造成了训练集。以下为构造成的训练集的样本情况展示，其中左边表示拥有该项，右边表示没有该项。蓝色代表正常软件的个数，红色代表恶意软件的个数。从图中可以看出，在某些权限调用上正常软件和恶意软件还是有着比较大的不同的，如读取系统信息，恶意软件95%都有着相关的权限，而正常软件却只有一部分拥有相关权限，又比如在系统内存占用率上，恶意软件普遍比正常软件需要更多的系统内存，占用更多的CPU资源。



J48决策树算法流程如下图所示:



本项目通过训练集的1000个样本由weka架包生成了j48决策树，接下来我们用下图生成的决策树对测试集进行分析。

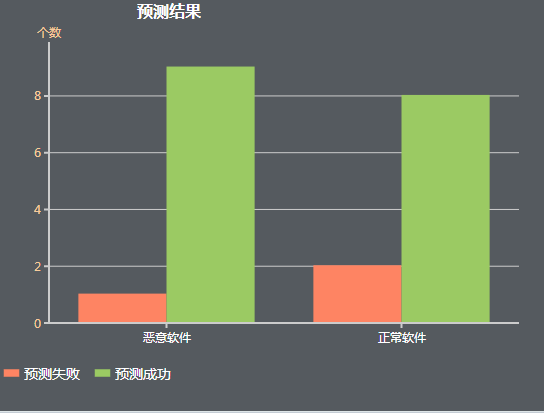


1. **决策树测试**

**小数据测试**

测试集由网上查找的20个项目通过上文中的aspectJ切面代码进行切面获得，其中包含10个正常软件样本，10个恶意软件样本。本次在github上传了4个项目。

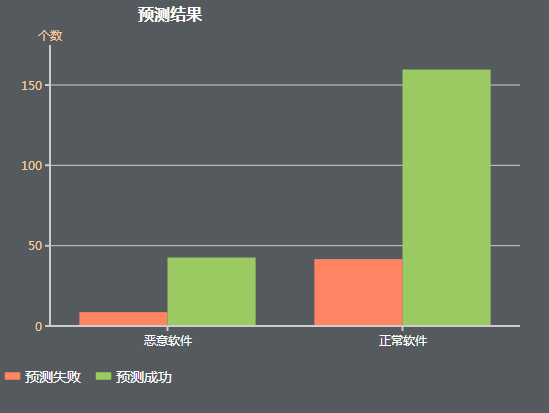
结果如图所示：



其中17个预测正确，三个预测错误，正确率为85%。

**大数据测试**

测试集由训练集中随机抽取200个用例组成，其中恶意软件20个，正常软件150个，最后分析的结果如下：



其中49个预测错误，201预测正确，正确率为80.4%。

**总结**

经过测试，本决策树模型的预测成功率大概维持在80%~85%之间，基本可以作为识别恶意代码的工具，满足识别恶意软件的功能要求，有效算法还有待优化以提高预测成功率，

参考文献

[1] Min L X, Cao Q H. Runtime-based Behavior Dynamic Analysis System for Android Malware Detection[J]. Information Technology Applications in Industry Computer Engineering & Materials Science, 2013，（6）：25-41.

[2] LI W Y. Research on Android Malware Detection Technology[J].Netinfo Security, 2015, (9) : 62-65.

[3] Manish Jain,Dinesh Gopalani.Testing applicat security with aspects. Electrical, Electronics, and Optimization Techniques (ICEEOT),International,Conference on.2016

[4] [胡文君](http://s.wanfangdata.com.cn/Paper.aspx?q=%e4%bd%9c%e8%80%85:"%e8%83%a1%e6%96%87%e5%90%9b"), [赵双](http://s.wanfangdata.com.cn/Paper.aspx?q=%e4%bd%9c%e8%80%85:"%e8%b5%b5%e5%8f%8c"), [陶敬](http://s.wanfangdata.com.cn/Paper.aspx?q=%e4%bd%9c%e8%80%85:"%e9%99%b6%e6%95%ac"), [马小博](http://s.wanfangdata.com.cn/Paper.aspx?q=%e4%bd%9c%e8%80%85:"%e9%a9%ac%e5%b0%8f%e5%8d%9a"). 一种针对Android平台恶意代码的检测方法及系统实现[J]. 西安交通大学报, 2013, 47(10).

[5] [Xiao Lu](https://www.engineeringvillage.com/search/submit.url?CID=quickSearchCitationFormat&implicit=true&usageOrigin=recordpage&category=authorsearch&searchtype=Quick&searchWord1={Xiao+Lu}&section1=AU&database=2&yearselect=yearrange&sort=yr" \o "Search Author)1,  [Long Peng-Fei](https://www.engineeringvillage.com/search/submit.url?CID=quickSearchCitationFormat&implicit=true&usageOrigin=recordpage&category=authorsearch&searchtype=Quick&searchWord1={Long+Peng-Fei}&section1=AU&database=2&yearselect=yearrange&sort=yr" \o "Search Author). The Research of AOP Based on Java Dynamic Proxy.MicrocomputerInformation, n2, 211-13, 2011;