通过自动化静态代码分析和机器学习对Android应用分类

Asaf Shabtai Yuval Fledel Yuval Elovici

德国电信实验室，信息系统工程系，本古里安大学

摘要

本文我们通过分析从Android应用安装包中提取中的静态特征，应用机器学习技术实现了Android应用的分类。这些静态特征从Android应用的Java字节码文件（即class.dex 文件）和XML等其它类型的文件中提取。我们的测试主要集中在将Android应用分为工具和游戏两类。工具和游戏两类应用的成功区分向我们传递了一个积极的信号。这表明我们如果以同样的方法对良性Android应用进行学习和建模以发现潜在的恶意Android应用具有一定的可行性。我们总共对2285个Android安卓包进行了测试。结果表明，从APK文件中提取的静态特征配合机器学习算法可以在不需要运行应用的情况下很好地检测出应用的内在特征。因此，这种技术很可能可以用来帮助检测恶意软件。

关键词：移动设备，机器学习，恶意软件，安全，安卓，静态分析

1. 简介

智能手机已经从简单的移动设备演变成复杂紧凑的小型计算机。作为开放、可编程的网络设备，智能手机非常容易受到病毒、特洛伊木马和蠕虫等恶意软件的威胁。利用智能手机用户可以浏览互联网，收发电子邮件和短信，与其他设备交换信息，使用各种应用程序，然而正是这些操作使设备成为了潜在的攻击目标。[1][2]

被感染的智能手机可能会同时给用户和运营商带来严重的损失。恶意软件可以让手机部分或全部的功能失效；造成不必要的计费；窃取用户隐私或感染用户的电话簿中的所有联系人[3]。用来攻击智能手机的媒介主要包括:蜂窝网络，互联网连接(通过WFI、GPRS / EDGE或3G网络)、USB和其他外围设备[4]。

近几年最常见的智能手机操作系统是Google的Android系统。Android智能移动设备是一个全面的软件框架，它包括一个操作系统，中间件和一组关键应用程序。我们对Android框架[5]的安全评估表明，恶意软件渗透到设备的事情不仅可能发生在Linux层，也可能发生在应用程序层(Java)，因此探索保护Android框架的方法是至关重要的。文献[6]-[8]中就提出了一揽子通用的移动设备安全解决方案和为Android定制的安全解决方案。

多年以来，PC端遭受到了大量的安全威胁[9]。如今，手机系统厂商也开始担心手机系统将面临的安全挑战。伴随着移动设备的普及，针对移动平台的安全攻击越来越多。因此许多安全厂商和研究者提出了各种的移动平台安全解决方案。例如，Symbian和Google设计他们的操作系统中，应用只能在特定的沙箱中运行，以最大限度的减小恶意软件传播的能力[10]。此外,常用的桌面安全解决方案也被移植到移动设备。例如，Botha等人[11]分析了常见的桌面安全解决方案评估了其在移动设备的适用性。当移植一个安全解决方案到移动设备时，如杀毒软件，我们应该考虑到移动设备资源(电源、CPU和内存)的有限性。除此之外，大部分杀毒软件的能力取决于已知的恶意软件特征库；因此当攻击者放出一个新的恶意软件时，杀毒软件并不能保护使用者。还有一些恶意软件可能针对特定的、相对少量的移动设备(如，获取机密信息或记录设备所有人的位置)，因此会持续相当长的时间，直到被发现。

Symbian操作系统中集成了一个健壮的应用程序签名和认证机制。虽然此机制主要依赖手工代码检查，但事实证明它有效地减少了恶意软件的攻击。苹果还要求所有应用程序通过严格的审查程序。然而，恶意软件作者仍然可以逃避手工代码检查[12]。因此，为了在摆脱手工代码检查，做出更明智的安全决策，研究人员正在寻找各种捕获应用程序语义的替代品。自从Android系统发布以来，几款静态分析Android代码的工具相继被开发出来。Schmidt等人[13]对一个Android代码静态分析框架进行了评估，并对本地应用的函数调用数据进行了分析。Chaudhuri [14]提出了一个正式的语言用来描述Android应用程序和应用程序组件之间的数据流。该语言可以用来静态分析Android应用程序和应用程序之间的数据流并将其和Manifest文件中定义的安全规范做比对。

本文中，我们提出了一种Android平台上检测未知恶意软件方法，即用机器学习的方法对从Android应用中提取出来的静态特征进行学习和建模。Android应用程序是以.apk结尾的归档文件，此结构非常类似Java中的.jar文件。和应用程序相关的代码和资源文件即被压缩在此文件中。Android应用的.apk文件中含有可以帮组我们理解应用程序行为的信息。这些信息主要包括应用所请求的权限、调用的方法和类、用户交互界面等。此项研究的主要目标是找出分类算法、特征选取方法和特征数量的最佳组合，以尽可能的提高检测未知恶意软件的准确率。以往大多数的研究都是从字节序列中提取特征，而本文，我们试着使用Android应用中一些有意义的特征，例如，应用的请求权限、调用方法和类。

以下两个事实支持了用此种方法检测恶意代码的可行性：1）上文中所提到的特征均可以从Android的安装文件中提取；2）上文中所提到的大部分特征不会因为代码被混淆而被隐藏。尤其的，Android应用的列表文件中的内容只在安装过程中使用，程序被安装后无法自主更改列表文件的内容。

本文的结构如下：在第二章，我们对前人的相关工作进行了一些调查。第三章，我们描述了我们的算法。然后，第四章我们进行了测试，并展示了测试结果。最后，第五章，我们进行了总结并讨论了未来的工作。

1. 相关工作

通过机器学习方法来实现自动化，通过扩展启发式的检测方法来达到检测未知恶意软件的目的。在这些方法中，我们用一些从程序中提取出来的静态特征来代表该程序。（例如，字节序列或者PE头特征）。分类器对代码特征进行学习和建模以对未知的程序文件进行分类。近期的研究表明，用这些方法进行分类可以达到非常高的准确率[15]。相比于在程序运行时从系统中获取特征，静态分析有很大的优势。一方面，静态分析耗费更少的资源和时间；另一方面，静态分析不需要运行相应程序。

在过去的九年里，有一些研究主要集中在个人电脑平台上基于二进制代码的未知恶意软件检测。文献[16]首次提出将机器学习的方法应用到基于二进制特征码的恶意软件检测上。他们从文件中提取三种不同类型的特征：程序头（Portable Executale section）,字符串特征(程序文件中会编码一些有意义的明文字符串，例如：”Window”，”kernel”，”reloc”等。)，字节序列特征，采用了四种分类算法：基于指纹的算法（反病毒），Ripper（基于规则的），本地贝叶斯，多重贝叶斯。该研究表明，所有的机器学习中的分类算法都比基于指纹的算法准确率高。Abou-Assaleh等人[17]介绍了一套用Common NGram (CNG)分类算法和 k-nearest neighbor (KNN)分类算法进行恶意软件检测的框架。首先为每个恶意和良性的软件创建一个代表性的描述，当有新文件来临时，通过和已知类型的所有描述信息进行比对，最后将其分派给最相似的那一类。Kotler 和Maloof[18]使用了一个包含1971个良性应用和1651个恶意应用的数据集。利用信息提取方法从文件中提取N元组合和500个特征。N元向量用二进制表示，N元组中的每个元素代表一个特征是有或者无。在他们的实验中，他们分别用IBK（KNN）、TFIDF、Naïve Bayes、SVM和Decsion Tree（J48）进行了建模。后三种算法也进行了相应的优化。本次实验中，效果最好的四种算法是改进J48、SVM、改进SVM和IBK。

最近，Moskovitch等人[19]公布了一项研究结果，该研究使用了超过30000个文件样本，从每个样本中提取一个基于字节序列的N元组特征。在其它研究中，样本的特征向量多是从指令序列中提取[20]。

1. 算法描述

用机器学习的方法对未知文件进行分类大体分为两个阶段：训练和测试。在第一阶段，我们会向系统提供一组已知分类的apk文件。根据我们预先定义好的特征表从每个文件文件中提取特征向量。然后把从训练集提取的特征向量和对应的分类作为学习算法的输入。通过处理和分析这些数据，学习算法会生成一个对应的分类模型。

接下来，在测试阶段，分类器会对一系列未知分类并且没有出现在训练集中的apk文件进行分类。测试集中的每一个文件都是第一次被分析，利用同样的规则，从每一个文件中提取特征向量。分类器会基于这些特征向量对测试集中的文件分类。我们通过分类器对测试文件的分类结果和文件的真实结果进行对比来评估分类器的性能。因此，我们也需要知道测试文件的真实分类。

由于目前还没有可用的Android平台恶意软件。我们通过用此方法区分出游戏和工具两类来证明此方法的可行性。工具和游戏两类应用的成功区分向我们传递了一个积极的信号。这表明我们如果以同样的方法对良性Android应用进行学习和建模以发现潜在的恶意Android应用具有一定的可行性。

此研究的目标是希望能够用数据挖掘的技术探索出能够精确检测未知Android文件的方法。我们对以下分类器的性能进行了评估：Decision Tree (DT) [21], Naïve Bayes(NB) [22], Bayesian Networks (BN) [23], PART [24],Boosted Bayesian Networks (BBN) [25], Boosted Decision Tree (BDT) [25], Random Forest (RF) [26], Voting Feature Intervals (VFI) [27]。

在机器学习的应用中，过量的特征会带来一系列的问题，例如：误导学习算法、过度拟合、增加模型复杂度。在准备阶段进行特征筛选可以帮助学习算法生成更加有效的，速度更快的检测器。不过，减少特征应该以保持检测器性能的准确性为前提。