**2018年全国大学生信息安全竞赛**

**作品报告**

**基于动静态结合分析的Android应用安全性多重检测模型**

**作品名称：基于动静态结合分析的Android应用安全性多重检测系统**

**电子邮箱：**

**提交日期：**

填写说明

1. 所有参赛项目必须为一个基本完整的设计。作品报告书旨在能够清晰准确地阐述（或图示）该参赛队的参赛项目（或方案）。

2. 作品报告采用A4纸撰写。除标题外，所有内容必需为宋体、小四号字、1.5倍行距。

3. 作品报告中各项目说明文字部分仅供参考，作品报告书撰写完毕后，请删除所有说明文字。(本页不删除)

4. 作品报告模板里已经列的内容仅供参考，作者可以在此基础上增加内容或对文档结构进行微调。

5. 为保证网评的公平、公正，作品报告中应避免出现作者所在学校、院系和指导教师等泄露身份的信息。

# 目 录

# 摘 要

近年来，Android系统智能设备在全球市场的占有率越来越高，Android系统的安全问题却层出不穷，一方面各大应用市场宽松的审查机制导致Android恶意软件泛滥，在网络上流传不止；另一方面开发者的水平不一也导致Android应用自身的安全性良莠不齐。随着时间的推移，各大厂商的手机大多都已安装了自家的安全软件，并配套了具有审查机制的应用市场。但是，由于Android系统的开源性，许多智能手机爱好者会为系统进行root提权，从而对手机进行高度定制。然而，各大厂商原生的安全机制对root手机不再适用，用户在没有对Android应用的安全性进行确认下，经过root提权的手机（下简称root手机）更容易被恶意软件以高权限入侵。与此同时，目前常见的Android应用安全性检测方案主要使用动态和静态两种检测技术，但大多数设计方案多只针对其中一项技术进行改进或直接应用，而少有对这些技术检测效果的相关性与互补性进行研究。这明显无法对root手机进行全面、有效的安全防护。

在此背景下，我们学习了现有的检测技术，在加以改进后进行综合应用，最终设计并实现了一种综合多种检测技术的Android应用安全性检测系统。通过不同检测方案间的互补性来设计检测流程，扩展检测的评估面，同时部分检测模块改进后的检测方案可以有效提高检测精度和性能。本系统具体工作包括：可以恶意应用进行有效拦截。具体工作包括：

（1）在Android设备上设计并实现动态检测相关的两个模块。动态检测通过捕获Android应用在运行过程中的系统调用特征和网络流量特征做判断，第一个模块基于Android API Hook技术进行敏感行为监控，并自行设计风险评估模型进行预警，另一个模块则基于代理和VPN设计实现了应用流量的监控。通过这两个动态检测模块，可以在恶意行为及恶意流量两方面对运行中应用实现实时预警，同时可以弥补由于加壳等原因导静态分析无法进行的问题

（2）在服务器端设计并实现静态检测相关的三个模块。第一部分通过提取权限信息与敏感API建立特征向量，结合对不同机器学习算法的性能评估，提出了一种基于随机森林算法的恶意软件鉴别模型，第二方面辅助以数据流分析技术实现高精度的漏洞扫描，第三方面采用FlowDroid提供的污点分析算法实现隐私泄露的检测。通过静态检测部分的三个模块，可以有效检测出恶意代码攻击，漏洞代码利用，隐私泄露三种问题，扩展应用安全性的评估面。

本文设计实现的动静态结合的Android应用安全性检测系统，经过测试，能够较好的处理应用在运行过程中可能存在的恶意流量、恶意行为、漏洞利用，隐私窃取等一系列威胁用户数据安全，降低软件安全性的行为，可为root手机的用户提供较为完善的应用安全保障，保护用户的信息安全。

# 第一章 作品概述

## 1.1 背景概述

Google在2007年11月发布Android系统以来，凭借着Android系统的开源性和高度的开放性，Android迅速占领移动终端市场。2016 年 IDC 发布最新数据显示，Android 全球市场份额超过 80%，并预计 2019 年 Android 市场份额将达到 82.6%，占据绝对优势[1]。另外，据互联网监测公司StatCounter的研究显示，在全球网络用户中，从移动设备访问互联网的用户数量在2016年11月首次超过桌面端用户。来自手机和平板电脑的网络流量已经合计达51.2%，超过了桌面端的48.7%。这种趋势已经不可逆转，促使越来越多的厂商向移动应用产业发展。随着移动应用行业的快速发展，在巨大经济利益的诱惑下，灰产和黑产纷纷盯上这块大蛋糕，导致各种恶意应用层出不穷。大致看来，Android应用有如下几大类影响软件安全性的问题：

1）应用自身含有恶意行为影响用户的信息安全。Android恶意应用通过推送恶意广告诱骗欺诈、私自下载其他应用偷跑流量、私发短信恶意吸费、出卖用户隐私等方式非法获取利益。更有甚者，在应用安装后下载木马后门程序，让用户的Android手机沦为“肉鸡”。产生这种现状的原因，一方面来源于国内各种Android应用市场宽松的审查机制，另一方面则源于Android系统自身的问题，正如Sundar Pichai 曾在MWC 大会上所说——“android 不是为了安全而设计的，他是为了开放而设计的”，这是这种开放的设计导致了Android平台恶意应用的肆虐。

2）应用自身编写存在问题而被恶意利用。由于应用开发者的水平参差不齐以及应用测试的不完善，导致了很多应用自身都存在安全问题，容易被攻击者恶意利用从而对用户产生威胁。要防护这类问题，还要从应用的安全开发做起。

3）应用过度的隐私收集。为了提高用户体验，或是提高广告投放的精准度，大多数应用都尽可能多的收集用户隐私，但作为移动设别普及的当下，用户的手机中存储者大量个人隐私，一旦应用过度收集用户个人数据，甚至是恶意窃取，就造成了隐私泄露问题。

关键词：Android，Hook，机器学习，静态分析，数据流，漏洞扫描，隐私泄露。

## 1.2 作品意义和特点

与PC端相对成熟的安全防护相比，移动安全领域还处于劣势，因此研究对抗Android恶意应用显得十分必要。目前Android恶意应用检测方法主要分为动态监控和静态检测[2]，这两种技术各有优缺点。静态检测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。静态检测技术有一个共同的特点：应用程序不被执行，它的优点是可以实现对应用程序所有功能的完整检查，但缺点同样也很明显，首先软件加壳，代码混淆，动态加载等技术可以有效防止应用被静态分析，无法准确获得应用内部代码的话，静态分析就无从进行，另一方面静态分析大多基于特征匹配，难以胜任未知行为的检测工作，因此需要另一种检测技术——动态监控的配合应用。动态监控的核心过程是在应用运行过程中动态监控应用的行为，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，基于这些动态运行时收集的特征来进行恶意行为的识别，可以有效应对动态加载，加壳等应用防护方案，相应的，其缺点则是无法保证对完整应用行为的监控，只要还有行为没有在用户收集上运行，就无法保证是安全的。综合看来，动静态检测各有其优缺点，在实际使用应结合应用。在本作品中，通过客户端运行的动态监控，以及服务端的静态检测来结合这两种技术，提供更为完善的安全防护。

另一方面，本作品中使用的检测手段都是已经推广使用开来并被证实有效的，但这些检测技术或多或少都还存在一些改进的空间，因此本作品的另一个着重点则是对作品中使用的检测技术进行改进，以达到更高的精确度与检出率。首先在敏感行为监控模块设计了风险评估算法，以识别应用运行过程中可能存在的敏感行为并进行实时预警；恶意软件鉴别模块则在敏感API作为特征向量的基础上进行微调，以提高恶意应用的检出率；漏洞检测模块中则结合数据流分析技术进行，使用过程内数据流分析技术设计了函数内变量行为记录算法来提高漏洞检测率。

# 第二章 作品总体设计

## 2.1 系统总体功能

## 2.2 系统工作流程

# 第三章 系统各模块详细设计与实现

## 3.1 动态检测部分

### 3.1.1 基于API Hook的敏感行为检测模块

### 3.1.2 Android流量监控、分析、拦截模块

Android恶意应用的获利方式几乎不能离开移动互联网，如果切断恶意应用的恶意流量，就可以阻止恶意应用进一步的恶意行为，从而保护用户的流量和隐私。

**3.1.2.1 Android流量监控方法**

拦截恶意流量的前提是获得Android网络通信流量。目前Android开发者通过下面几种方法获得经过Android设备的网络报文。

1. 使用代理服务器

开发者在Android设备中运行代理服务器程序，绑定特定端口，设置当前网络环境的代理地址为127.0.0.1，端口为代理程序绑定端口。一般情况下Android手机的网络环境为无线WLAN或蜂窝移动网络。无线WLAN环境下进入系统无线设置，填写代理地址和端口，保存即可。蜂窝移动网络环境下需要新建一个接入点APN，修改APN的代理和端口，选择这个新建的APN连接。正确设置网络环境的代理后，Android系统将整个系统流量交给代理服务器，开发者在代理服务器中就可以得到经过Android系统的网络报文。



图 Android代理服务器

常用的代理服务器有FiddlerCore、browsermob-proxy等。FiddlerCore是fiddler流量分析软件的嵌入式集成模块。开发者在自己的应用中集成FiddlerCore ，使用它提供的API即可获得fiddler的功能，如抓包过滤等。browsermob-proxy是一个开源的代理工具，提供了一套接口实现对网络流量的控制。这两种代理服务器都能够在用户信任代理服务器证书的前提下通过中间人攻击mitm的方式解析Https流量，获得加密通信的内容。

1. 基于VPNService编写VPN程序

Google在Android API level 14中引入了VPNService，为开发者提供了开发Android VPN程序的框架和接口。Android开启VPN后，VPN程序可以获得几乎所有网络流量。对于一个向外发送的数据包，一般会在Android VPN服务框架中经历4个转发过程。进入Android的数据包过程与之类似。



图 数据包在VPN中的转发过程

1. 应用程序使用socket，将数据包发送到真实网络设备上。
2. Android系统通过iptables，使用NAT，将所有的数据包转发到TUN虚拟网络设备上去，端口是tun0。
3. VPN程序通过打开/dev/tun设备，并读取该设备上的数据，可以获得所有转发到tun虚拟网络设备上的IP包。因为设备上的所有IP包都被NAT转成原地址是tun0端口发送，所以VPN程序可以获得进出该设备的几乎所有的数据，只有如本地回环数据等少数流量无法获得。
4. VPN程序可以对拿到的数据包做一些处理，然后将处理过后的数据包，通过真实的网络设备发送出去。为了防止发送的数据包再被转到tun虚拟网络设备上，VPN程序所使用的socket必须先被明确绑定到真实的网络设备上去。

Android一款名为packet capture的应用使用VPNService实现了流量抓包，此外开发者助手中也实现了VPN抓包的功能。

1. 使用Android tcpdump

Android tcpdump是一个命令行窗口工具集，可以抓取WLAN、蜂窝网络等环境下的数据流量。tcpdump是Linux下一款强大的网络数据采集分析工具，而Android tcpdump则是tcpdump针对ARM架构处理器的编译版本。tcpdump通过libpacp来抓取报文，在Linux中libpacp使用AF\_PACK套接字实现对流量的抓取分析。Android设备必须取得root权限才能使用tcpdump。

1. 改造Linux防火墙[3]

Andriod系统基于Linux，可借助现有的Linux防火墙，使用iptables/Netfilter链钩子技术，劫持系统调用，在内核中获取数据包并隐藏劫持行为，实现Android内核层次的流量控制功能。

1. 借助PC端软件

PC端具有功能强大的流量分析软件，如wireshark、fiddler、charles等。一般方法是PC端开启一个热点，待分析Android设备连接这个热点，PC端流量分析软件监听这个热点即可获得Android网络通信流量。

**3.1.2.2 流量监控方法的优缺点分析**

1. 基于代理服务器的方案中可用到的FiddlerCore提供了可供开发者二次开发的API，而browsermob-proxy是优质的开源项目，并且Android设备仅需设置代理IP和端口即可，因此实现较为容易。局限性是代理服务器只能在应用层上进行流量监控和拦截修改，无法在传输层上进行，因此不能区分开不同Android应用的流量。
2. 使用VPNService框架编写的VPN程序可获得Android设备中网络层的IP报文，因此可通过IP报文中应用层的端口信息区分出不同应用的流量并对流量进行控制，功能更为强大。局限性是VPN程序需要对IP数据包进行不同协议下的封包拆包操作，需要大量底层网络报文控制代码的编写，实现较为困难。
3. 改造Linux防火墙，在内核级取得对Android流量控制。这种方式用户可以不进行任何操作如设置代理、开启VPN等，就可以获得所有流量。局限性是需要研究和修改系统源码，实现比较困难。
4. Android tcpdump命令行工具不适用于嵌入Android应用程序中。此外tcpdump只能抓取流量，无法拦截修改流量，因此tcpdump无法用于本模型。
5. 建立热点借助PC端软件的方案是分析人员临时分析Android流量时最方便最有效的方法。若使用fiddler，可在应用层监控、拦截、修改报文；若使用wireshark，可监控链路层以上的所有报文，但不能拦截修改报文。局限性是只能间接分析使用。

**3.1.2.3 设计目标和功能**

本模块搭建一个Android流量监控、分析、拦截平台，实现以下4点功能。

1. Android设备上流量抓包，根据应用、进程分类显示。
2. Android流量拦截功能。建立URL模式规则库，根据规则库中的黑名单自动拦截恶意流量。
3. 获取hook模块输出的可疑URL，由用户授权决定是否进行拦截。
4. 用户自定义拦截规则，服务器维护共享规则，同步优质规则。

鉴于上述功能的需要，考虑现有Android流量监控方法的优缺点，对于流量抓包功能，向用户展示具体每个应用的流量是抓包的最终目的，所以流量抓包功能采用VPN方法实现；对于流量监控拦截功能，不必展示详细的信息，而browsermob-proxy开源项目除了提供较为完善的接口外，开发者还能修改源码实现更特殊的需求，所以流量监控拦截功能采用browsermob代理实现十分合适。

**3.1.2.4 Android流量监控、分析、拦截平台的实现**



图 Android流量监控、分析、拦截平台架构图

Android流量监控、分析、拦截平台采用C-S模式，Android客户端实现流量的抓包和拦截，服务器实现共享规则的汇总、选优、同步。

1. 客户端的设计

Android客户端中流量分析模块采用VPNService实现。用户选择想要分析的应用，启动VPN程序，VPN程序启动完毕后，再启动想要分析的应用，之后待分析应用的所有流量都会被VPN程序获得并显示。VPN程序拿到每一个IP数据包，检查其中是否包含了传输层协议，如果是，根据传输层中TCP或UDP中的端口信息将IP数据包与待分析应用绑定。



图 IP数据包与特定应用绑定流程

Android中通过Runtime.getRuntime().exec()方法可以执行Linux的命令，从返回结果的Process中获得输出流，从输出流中获得Linux命令的执行结果。通过上述方法，执行ps命令获得进程的PID和NAME，执行cat/proc/net/tcp、cat/proc/net/udp获得进程PID和绑定的运输层端口。两个命令的输出通过PID联系起来，可将进程名NAME和进程端口对应，从而区分每个IP数据包到底是哪个应用程序的。

将IP数据包与应用程序对应后，解析IP数据包的传输层以及应用层，将数据包的基本信息如源和目的IP地址、端口、应用层协议等展示给用户。通常情况下应用传输的数据会使用Http或Https协议，抓包程序突出显示Http键值对和图片等信息，更加方便地让用户追踪应用行为的流量信息。

Android客户端中的流量拦截模块采用browsermob-proxy代理实现。browsermob-proxy使用是为ajax应用和selenium测试开发的代理服务器，由于browserm-proxy完全使用java编写，可方便移植到Android平台上运行。为了支持对Https流量的解析，修改代理的证书部分并安装代理服务器的证书，以支持Android平台上的MITM。

图 MITM中间人攻击解析Https流量

Https协议在传输层之上增加了TLS/SSL层，对应用层明文信息进行加密，普通抓包方法只能得到加密后的密文。MITM中间人攻击方式作为本地Android应用和应用对应服务器的中间人，使用用户信任的代理证书，充当本机应用的服务器又充当应用远端服务器的客户端。代理在这过程中可从两个SSL连接中解密得到明文，实现对Https的解密抓包。

客户端中使用SQLite存储恶意URL模式表。代理根据这个数据库匹配每个从Android设备中发出去的请求报文的URL，如果命中了规则，代理就将请求报文拦截并向应用返回内容为空的回应报文，从而切断远端服务器发来的恶意请求。如果请求报文没有命中任何规则，就将此请求报文放行至互联网。



图 恶意URL正则模式表结构

由于代理必须检查所有从Android设备中发出去的请求报文，在网络负载较大的情况下，要求代理的拦截部分具有较高的效率，才能保证不明显影响网络通信速率。代理判断一个请求报文的URL是否为恶意的，必须要保证这个URL不能命中任何一个恶意规则库中的正则表达式，这相当于对每一个请求都会对数据库进行一次遍历。影响效率的主要因素是数据库的访问比较耗时，为了平衡访问数据库的次数和运行内存的占用，采用了折衷的方法。



图 代理拦截URL匹配流程图

每次访问数据库都取出X条记录，参数X需要根据数据库的规模动态调整。假设数据库规模较小，完全可以取出数据库中所有记录放入内存，匹配速度最快。随着数据库规模的增大，X可取数据库总记录数的二分之一、四分之一等等。这样可以减小内存的占用，控制访问数据库的次数，提高匹配效率。

当Hook模块判断当前用户使用的应用为恶意应用，Hook模块将恶意应用的运行日志传入流量拦截模块。拦截模块使用正则表达式提取Hook日志中恶意应用曾经向外发出的所有流量的URL，添加到恶意URL黑名单中，拦截标志全部设为true，代理默认全部拒绝放行。



图 恶意URL黑名单结构

用户如果需要继续使用恶意应用的某些功能，选择信任该应用，拦截标志变为false，代理则放行流量。用户后续能够修改拦截标志字段限制恶意流量。

用户可使用自定义规则模块操作数据库，自由控制代理的拦截行为。Hook模块提取的日志和抓包模块的结果为用户提供数据流量信息，用户编写的自定义规则存放在本地恶意规则库中，结构与恶意URL模式表相同。在用户同意分享自定义规则的条件下，平台定期上传用户自定义规则到服务器。

1. 服务器的设计

服务器收集汇总客户端分享的用户自定义规则，存放到服务器端的共享恶意URL模式数据库，向所有用户开放。用户可以尝试共享的规则，并对其进行评分评价。服务器定期对具有良好口碑的高评价规则添加到公共恶意URL模式数据库中，定期向所有客户端推送更新。随着用户共享的优质恶意URL规则的积累，逐步加强对恶意流量拦截的广度和深度。

## 3.2 静态检测部分

### 3.2.1 基于权限与敏感API的机器学习分类算法静态检测模型

### 3.2.2 Android应用漏洞检测模块

### 3.2.3 隐私泄漏检测模块

# 第四章 系统测试与分析

## 4.1 测试环境

## 4.2 测试方案

## 4.3 测试过程

## 4.4 测试结果总结

# 第五章 总结与展望

## 6.1 工作总结

## 6.2 未来展望

# 参考文献

[1]陈鹏,赵荣彩,单征,韩金,孟曦.基于动静结合的Android恶意代码行为相似性检测[J].计算机应用研究,2018,35(05):1534-1539.

[2]文伟平,梅瑞,宁戈,汪亮亮.Android恶意软件检测技术分析和应用研究[J].通信学报,2014,35(08):78-85+94.

[3]刘淼. Android流量监控技术研究与实现[D].华中科技大学,2012.