基于Android内核驱动的白名单网络控制

杨易达[1],孙钦东[1,2],胡国星[3],李元章[3*]
(1. 西安理工大学计算机科学与工程学院,陕西西安 710048;2. 西安交通大学网络空间安全学院,陕西西安710049;
3. 北京理工大学计算机学院,北京100081)

摘 要: Android系统是目前主流的移动终端操作系统之一,其数据泄露问题日益受到学术界的广泛关注.恶意应用窃取用户敏感数据后通过互联网发送扩散,从而对用户实施进一步侵害.Android系统中网络权限属于常规权限,应用无需用户授权即可联网发送数据.针对上述问题,本文提出了一种基于Android内核驱动程序的网络白名单网络控制方案,用户可以监控所有应用程序的网络使用状态,选择信任的应用加入白名单中,对白名单中的应用程序实行内核级签名验证,防止程序代码被非法篡改,从而构建安全可控的网络使用环境.本方案为应用和内核的通信构建了专用通道,以确保网络白名单管理权限不会被其他应用窃取,随后通过进程识别针对性地管控网络权限,在不影响正常应用功能的情况下实现权限管理.经过实验验证,本方案可以有效防止恶意应用利用互联网泄露用户隐私,网络管控成功率达到了100%.系统运行稳定,被管控应用启动时间最大增加33.1%,最小增加3.6%.

关键词: Android;网络白名单;数据泄露;进程识别;网络控制
基金项目: 国家自然科学基金(No.62072037, No.U1936218)
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112(2024)03-0967-10
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20220611

A Whitelist Network Control Based on Android Kernel Driver

YANG Yi-da[1], SUN Qin-dong[1,2], HU Guo-xing[3], LI Yuan-zhang[3*]
(1. *School of Computer Science and Technology, Xi'an University of Technology, Xi'an, Shaanxi* 710048, *China*;
2. *School of Cyber Science and Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi* 710049, *China*;
3. *School of Computer Science and Technology, Beijing Institute of Technology, Beijing* 100081, *China*)

Abstract: Android system is one of the most popular mobile terminal operating systems at present, and its data leakage problem has been increasingly concerned by the academic community. Malwares steal users' sensitive data and spread it over the Internet to harm users further. In the Android system, network permissions belong to common permissions, and applications can send data by internet without user authorization. To solve the above problems, this paper proposes a network whitelist scheme to control network based on Android kernel driver. Users can monitor the network usage status of all applications and select trusted applications to be added to the whitelist, and a kernel-level signature verification for applications in the whitelist is applied to prevent illegal tampering of execution code, thereby creating a safe and controllable network usage environment. A dedicated channel is constructed for the communication between applications and the kernel to ensure that the network whitelist management permissions are not acquired by other applications. Then, the network permissions are controlled through process identification to achieve permission management without affecting normal application functions. Through experimental verification, this scheme can effectively prevent malwares from using the Internet to leak users' privacy data, and the success rate of network control has reached 100%. The system runs stably and the startup time of controlled applications increases by a maximum of 33.1% and a minimum of 3.6%.

Key words: Android; network whitelist; data leakage; process identification; network control
Foundation Item(s): National Natural Science Foundation of China (No.62072037, No.U1936218)

收稿日期:2022-05-27;修回日期:2022-08-17;责任编辑:梅志强
*通讯作者:李元章

1 引言

Android自诞生之日起,全球市场占有率迅速上升,于2011年首次超过Symbian系统[1],之后更是持续飙升.据statcounter的数据显示[2],截至2021年7月,在全球手机操作系统中Android的市场份额达到了72.27%,稳居第一.随着Android市场份额的增多,除了带来巨大的用户量以外,还必然会带来很多安全问题[3~5].虽然Android在诞生之日起就吸收了其他操作系统的优点,在安全方面下足了功夫,但由于其完全开源的特性,以及用户的安全意识还不够高,时至今日,Android依然存在着许多安全隐患.移动安全公司Zimperium在2022年发布的全球移动威胁报告指出[6],Android在2021年发现了574个安全漏洞,其中135个安全漏洞的安全评分分数高于7.2,18个安全漏洞被评为严重威胁.

Android面临的安全威胁更多地是用户隐私数据的泄露[7].具体表现为:(1)用户在连接不安全的热点,扫描恶意二维码,或者点击恶意网站链接时,攻击者可以劫持用户的网页,令用户在进行网页浏览时跳转到钓鱼网站,从而在用户毫无防备的时候获取其输入的账号和密码[8];(2)恶意应用程序在使用时权限越界,要求用户授予其一些额外权限,例如读取通讯录、读取位置信息、录音、拍照等,而如果用户缺少安全意识,就给了恶意应用程序收集用户信息的机会;(3)一些正常应用程序虽然没有权限越界,但由于应用程序在编写时存在安全漏洞,且其正常工作时需要用到敏感权限,这就容易被攻击者利用,从而导致数据泄露[9].

本文在Android系统中针对恶意应用利用互联网泄露用户数据的行为,研究了一套网络白名单系统.普通应用无法在应用层对其他应用进行网络管控,本方案结合内核的设备驱动实现了该功能.同时本章构建了应用和内核的专用通信通道,避免指令被其他应用使用.用户可以在本系统的管控应用中监控其他应用的网络使用情况,并选择信任的应用加入白名单中,实现对应用的网络管控.

本文的主要创新点与贡献如下:

(1) 构建了应用专用通信通道.系统调用是应用使用内核服务的主要方式,但是缺少权限验证.本文提出了在应用的专属存储空间中搭建专用通信通道的方案,可以为应用提供独有的和内核通信的方式.

(2) 对内核的驱动层进行改造以实现网络管控功能,不影响其他层的正常运行,对系统的性能损耗小,使用方便,运行稳定.

2 相关工作

2.1 Android系统架构

本文旨在构建网络可控的Android环境,因此深入了解Android系统架构非常必要.Android系统架构由底至上分为七层[10~13],分别是Loader层、Linux内核层、硬件抽象层、系统库层、系统服务层、Binder IPC层和应用层.

Android是基于Linux内核开发的,而Linux是一种基于POSIX(Portable Operating System Interface of UNIX)的操作系统,处于系统架构底层的位置[14],提供和硬件的交互,不直接参与用户交互.Linux内核层主要支持各种厂商的驱动程序[15],如相机驱动、音频驱动、显示驱动等.Bootloader加载后的第一个任务就是运行Kernel代码.Linux内核层的特殊位置使该层具有较高的权限,因此在本文的研究方案中一些权限较高的操作将通过修改Linux内核代码完成.

2.2 APK应用程序的完整性校验

APK(Android application Package)签名可以保证APK的完整性和来源的真实性.签名过程首先计算APK内容的哈希值,再对哈希值使用签名者的私钥进行签名.校验过程则通过签名者的公钥对签名的密文进行解密,再计算APK内容的哈希值,获得的明文与哈希值进行比对,二者一致则校验通过.

Android系统会对安装应用包中的DEX(Dalvik Executable)文件进行优化和编译,Android O版本之后的系统可以先将DEX文件优化成VDEX(Verified DEX)文件,VDEX文件中包含解压后的DEX文件和一些用于加快验证速度的元数据.而VDEX文件中的部分模块又会被提取并经过AOT(Ahead-Of-Time)编译成直接可以执行的ODEX(Optimized DEX)文件,优化过程中生成的VDEX和ODEX文件来说,这些文件并不存在于APK安装包中.

为了对应用程序进行完整性校验,仅仅检查APK内容是不够的,VDEX、ODEX文件包含了应用程序运行时的指令代码,需要加以验证.

2.3 Android网络架构

Android的网络部分分为网络协议接口层、网络设备接口层、设备驱动功能层和网络设备与媒介层.各层的作用如下:

(1) 网络协议接口层:不论网络层协议是IP(Internet Protocol)还是ARP(Address Resolution Protool),都向上层提供统一的数据包收发接口.

(2) 网络设备接口层:该层用一个结构体net_device描述具体网络设备的属性和操作,提供给网络协议接口层,同时也作为设备驱动功能层中各函数的容器.该层将网络设备封装,实现了宏观上对网络设备结构的描述.

(3) 设备功能驱动层:驱动提供了操作硬件的逻辑方法.该层驱动网络设备硬件完成数据的发送和中断

接收操作,实现了net_device结构体的具体功能.

(4) 网络设备与媒介层:该层包含网络适配器等用来发送和接收数据的物理实体,通过设备功能驱动层进行驱动.

设备功能驱动层是数据流动层面的最底层,再之后就交由物理设备进行数据的发送和处理. 因此,网络白名单方案适合在设备功能驱动层对网络传输操作进行识别过滤.

2.3 网络抓包技术

为了分析网络白名单的实际效果,需要对手机发出的网络数据包进行捕获,从而判断网络管控功能是否生效以及是否存在漏洞. 网络抓包的原理是监控客户端和服务端之间的某个网络节点,获取所有经过节点的数据[16],并解析这些网络协议. 但是由于中间的网络节点不受控制,无法在中间节点进行抓包,所以常用的抓包方式是在客户端和服务端进行抓包. 图1为手机在Wi-Fi环境下和移动网络环境下的抓包示意图.

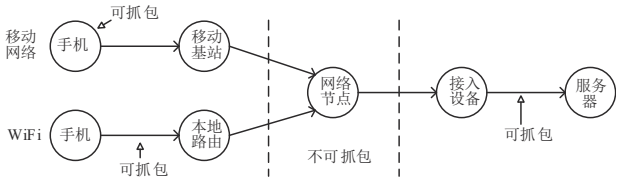


图1　手机网络抓包原理

本文的方案中,服务器端为各个应用程序提供,无法进行抓包. 因此本文选择在客户端抓包. Wi-Fi环境下可以在手机发给路由器的过程中加上一层代理,比如用计算机开启热点供手机连接,此时计算机就是数据包从手机发往路由器的代理,在计算机上即可抓取所有手机发出的数据包. 移动网络下手机数据直接发送给移动基站,需要在手机处直接进行抓包,然后将抓取到的数据发送到计算机端进行分析,这种抓包方式不够直观,使用起来较麻烦. 常用的抓包工具有wireShark[17]、tcpdump[18]、Charles、mitmproxy等. 这几个工具的对比如表1. wireShark和Sniffer都是综合性的跨平台抓包软件,wireShark更适合在计算机端利用图形界面实时分析抓包情况,而tcpdump由于可以用命令行在后台运行,更适合在手机端和服务器进行抓包,Charles和mitmproxy则是专门用于http协议的抓包工具.

3 网络管控和指令通道

3.1 功能分析

网络白名单需要实现两个功能:为用户构建一个方便的白名单管理界面和赋予白名单里的应用使用网络的权限.

表1　网络抓包工具对比

工具	跨平台	界面操作	命令操作
wireShark	是	是	否
tcpdump	是	否	是
Sniffer	是	是	否
Charles	是	是	否
mitmproxy	是	是	是

本文开发了一款应用作为网络白名单管理界面,为方便叙述,称为管控应用. 在管控应用上用户可以实时监测其他应用对网络的访问情况,并可以勾选信任的应用加入到白名单中. 而对于网络管控模块,由于应用与应用之间无法互相限制网络访问权限,因此无法在应用层完成此功能. 最终本文选择在内核层实现网络管控模块,原因是内核层权限高,可以检测并截获所有进程使用网络的请求. 网络白名单功能框架如图2所示,红色虚框部分为需要实现的内容. 管控应用生成白名单,内核接收管控应用发送的指令,根据指令内容完成网络管控、白名单更新等任务. 该方案的关键是网络管控模块和指令系统如何实现.

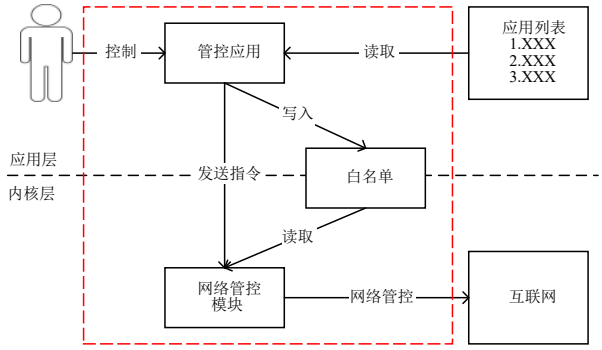


图2　网络白名单功能框架

3.2 基于驱动的Wi-Fi与modem管控

手机通过Modem(调制解调器)和Wi-Fi两种模式接入移动互联网. 未使用Wi-Fi时手机直接通过Modem将数据发往基站,由基站发往互联网. 使用Wi-Fi时,手机通过内置Wi-Fi天线将数据发往路由器,由路由器完成互联网接入. 网络管控模块需要对这两种联网模式进行管控.

为了对Modem和Wi-Fi进行管控,通过实验验证,所有的数据发送都要经过驱动传送到设备,在驱动层进行网络管控能保证传输检测的完备性,杜绝应用从别的渠道发送数据. 而Modem和内置Wi-Fi作为硬件

必须使用驱动程序驱动,因此本文选择在驱动层实现网络管控模块.

3.3 指令专用通信通道

网络白名单方案中通信数据的流向只有应用层到内核层,因此只要考虑管控应用向内核发送指令的情况.应用一般无法直接和内核进行通信,如需使用内核提供的功能时,正常的流程是先向用户申请授权,随后将请求发送给Android提供的系统服务,由系统服务在框架层统一调用内核提供的系统调用函数.如果在内核中增加了有关网络白名单功能的系统调用,这些系统调用不是管控应用所独有,存在被其他应用使用的风险.

本文设计了一种专用通信通道供管控应用和内核通信.Android系统下每个应用独立运行在虚拟机中,虚拟机会为其在目录“/data/user/0/应用包名/files”下申请一段存储空间.应用只能访问自己空间内的数据,不同应用之间除了进程间通信外,无法直接访问其他应用的存储空间.而内核的权限则相对较高,几乎可以对任何文件数据进行读取和修改.基于这个特性,可以针对性的将指令设计在管控应用的存储空间内,从而防止其他应用使用网络管控指令.应用对存储设备的任何操作,都可以在内核中截获,通过判断操作路径是否在管控应用存储空间内来确定操作目标是否具有网络管控权限,若具有网络管控权限则进一步判断该操作是否为指令操作,从而建立专用通信通道.如图3,本方案截获新建文件操作,用文件路径识别管控权限,用文件名传递指令.另外,将记录网络白名单的文件及其哈希值也存储在管控应用的存储空间内,可以防止其他应用修改白名单.

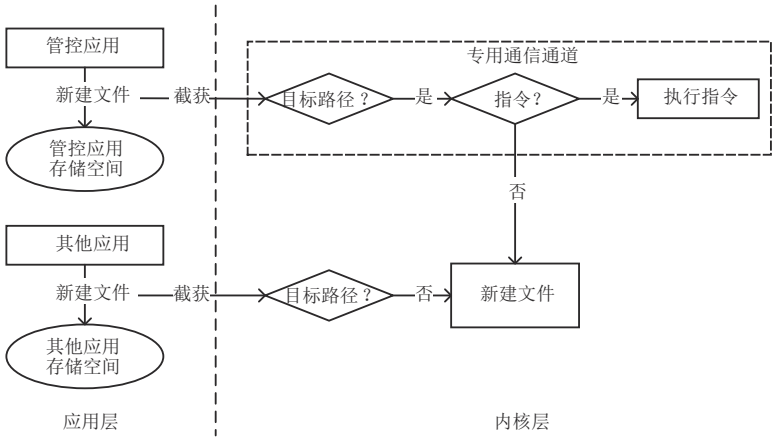


图3 专用通信通道

4 网络白名单方案

本节详细介绍了网络白名单方案的研究过程.首先介绍了网络白名单配置文件、指令系统和管控应用的设计,随后在内核中对网络白名单功能进行了适配,最后解决了网络白名单方案中出现的80端口问题.

4.1 网络白名单配置文件

网络白名单配置文件中记录了允许使用网络的应用列表,而用来表示应用的标识应满足以下两个要求:

(1)具有稳定性和唯一性.稳定性是指该标识不会因手机重启,应用重新安装或其他原因而改变.唯一性是指,通过这个标识只能对应一个应用,不存在多个应用共用一个标识的情况.

(2)内核中网络管控模块可以通过这个标识管控目标应用的网络权限.

应用安装到Android系统时会分配一个用户id(uid),用来在系统中标识这个应用.应用在使用网络时会以一个或多个进程的方式调用网络模块,多个进程之间的共同点是uid相同.因此在网络管控模块中可以通过uid来判断该进程是否具有权限.但应用卸载后,uid会释放,再次安装会分配新的uid,因此uid作为应用标识不具备稳定性.包名(Package name)是Android应用的唯一标识,Android系统里不能同时安装两个Package name相同的应用,这就保证了Package name对应应用的唯一性.而稳定性方面,Package name在应用开发时就被确定,除非重新开发,否则不会改变.Package name和uid的对比如表2.

uid和Package name通过路径/data/system下的packages.list文件进行关联.应用安装时系统会将分配

表2 Package name与uid对比

属性	稳定性	唯一性	作用范围
Package name	不变	唯一	应用层
uid	应用重新安装后改变	唯一	内核层

的uid和Package name生成一条记录写入packages. list中. 本文在白名单配置文件中记录允许使用网络的应用Package name,在内核中读取这些Package name并通过packages.list转化为uid,作为网络管控模块对应用进行管控的判断依据.

4.2 VDEX及ODEX文件校验

在内存管理中,每个进程拥有一个独立的虚拟地址空间,内核为每个进程都维护了一个页表,通过页表完成虚拟地址到物理地址之间的转换. 进程中的内存地址空间通过VMA(Virtual Memory Areas)的结构来管理,每一个VMA即代表虚拟内存空间中的一段连续区域. 而mmap系统调用通过初始化VMA结构,从进程的虚拟地址空间中分配一段连续的内存区域,并为内存区域创建一个映射关系. Android系统中的进程在启动时,会执行mmap系统调用将可执行文件映射到虚拟地址空间,进程直接读写内存来访问文件数据,减少数据拷贝次数,提高执行效率.

执行mmap系统调用时,首先切换到内核态,依次调用mmap_pgoff和ksys_mmap_pgoff等函数,最后由mmap_region函数清除旧的VMA,和其它虚拟内存区域合并,同时创建描述虚拟内存区域的结构体vm_area_struct,返回内存映射在内存空间的起始地址.

因此,本方案在ksys_mmap_pgoff函数中,执行对VDEX及ODEX文件的校验,判断应用程序是否被非法修改.

ksys_mmap_pgoff函数的参数包括待创建映射的文件描述符、文件偏移、内存映射的大小和映射标志,内核中的扩展代码通过文件描述符,结合当前的进程名,获取VDEX及ODEX文件的哈希值,与计算出的哈希值进行比对,二者一致说明应用程序未进行修改.

4.3 管控指令及其实现

管控应用发送给内核的指令内容如表3所示. 其中80端口管控问题在第4.5节中进行介绍.

表3　网络白名单指令内容

指令	描述
white_list_on	网络管控功能开启
white_list_off	网络管控功能关闭
white_list_update	更新白名单
portcontrol_on	80端口管控开启
portcontrol_off	80端口管控关闭

根据3.3节对指令专用通信通道的分析,指令系统的实现方案是,在内核/Kernel/fs/open.c的do_sys_open()函数中截获新建文件的操作,对该操作的目标文件路径和文件名进行判断,其中文件路径作为验证信息,文件名作为指令信息. 验证通过后根据不同的指令执行不同的任务,根据指令的内容按照表3执行对应的管控功能,例如获取的指令为white_list_on,则根据Package name所对应的uid开启该应用的网络管控功能,如果获取的指令为white_list_update,则进行更新白名单操作.

本文的管控应用Package name为com.whitelist.net. Android系统会在路径/data/user/0/com.whitelist.net下为管控应用生成存储空间. 管控应用的设计实现有如下四点:

(1) 应用启动时检测是否获得了用户存储权限和应用使用流量数据的访问权限,若没有权限则提示用户授权.

(2) 通过Appinfo类读取手机安装的应用列表,并通过TrafficStats类获取其他应用的流量使用情况,从而实时监控其他应用的网络使用情况.

(3) 用户勾选进入白名单的应用列表后,通过Appinfo类获取这些应用的Package name,写入/data/user/0/com.whitelist.net/white_list配置文件内,最后发送white_list_update和white_list_on指令,即在存储空间内新建文件名是white_list_update和white_list_on的文件.

(4) 用户点击关闭白名单功能按钮时,发送white_list_off指令;用户点击开启白名单功能按钮时,发送white_list_on指令;用户点击关闭80端口管控按钮时,发送portcontrol_off指令;用户点击开启80端口管控按钮时,发送portcontrol_on指令.

4.4 内核适配白名单

内核层在启动时读取白名单配置文件中的Package name到内存,并通过packages. list转换为uid. 这种设计思想为用空间换时间,即先把需要用到的uid存储到内存中,之后使用的时候直接从内存读取而不用从硬盘读取,可以提高网络管控模块的运行效率,降低系统性能损耗. 在内核中设置whitelist_flag和port80_flag两个标志位,分别用以表示网络白名单功能和80端口管控功能的开启和关闭状态. 将Package name转换成uid后,内核根据whitelist_flag和port80_flag的状态在Modem驱动和Wi-Fi驱动下进行网络管控. 在接收到white_list_update指令后,内核重新执行读取白名单配置文件,转换uid的流程,以达到实时更新白名单的目的;接收到white_list_on、white_list_off指令后对whitelist_flag进行修改;接收到portcontrol_on、portcontrol_off指令后对port80_flag进行修改. 图4为内核适配白名单框架.

对Modem驱动的网络管控操作在Kernel/drivers/misc/modem_v1/modem_io_device. c中的vnet_xmit()函

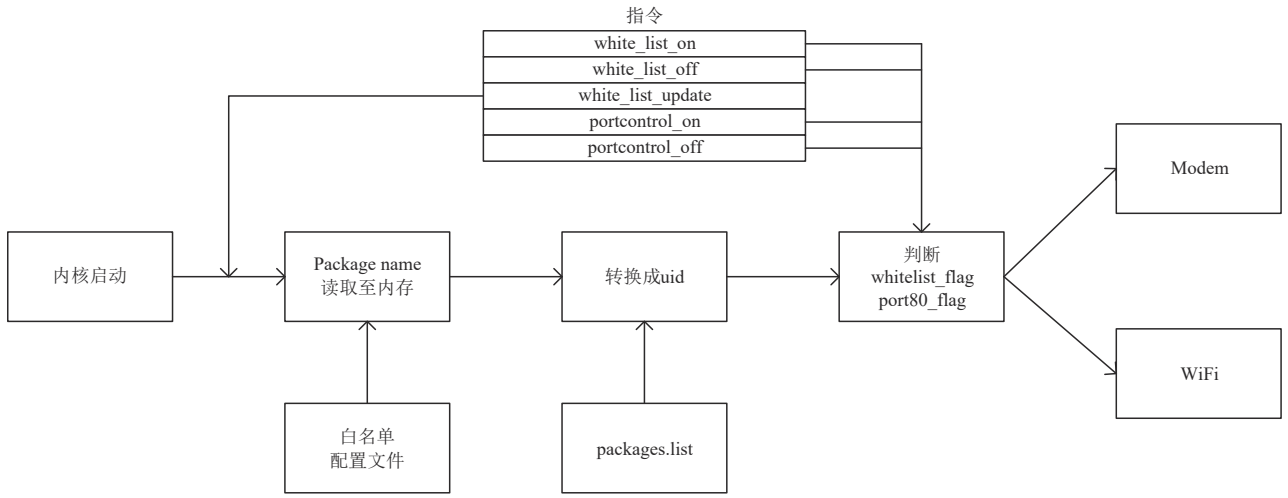


图4 内核适配白名单框架

数中完成，对Wi-Fi驱动的网络管控操作在Kernel/drivers/net/wireless/broadcom/bcmdhd_100_15/dhd_linux.c中的dhd_start_xmit()函数中完成. 在这两个函数里判断当前进程的uid是否在白名单里，如果是，则正常执行，不是，则返回设备不存在，阻止进程联网.

4.5 80端口问题

研究发现，uid为0或1 000且pid(进程号)>0的进程是一些系统进程，其主要功能为通过联网进行时间校准，位置同步等. 因为这些进程均通过端口号为80的端口进行联网，因此将其称为80端口管控. 考虑到恶意应用可能会通过系统进程进行联网并转发数据，网络白名单方案默认禁止系统进程联网. 当用户需要系统进程联网时，点击管控应用中的关闭80端口管控按钮即可.

5 测试与评估

本节分别在Wi-Fi环境下和移动数据环境下对网络白名单功能进行测试，最后分别评估了系统和应用的性能情况.

5.1 测试环境

网络白名单的测试环境如表4所示.

5.2 uid和Package name的对应关系

应用安装时系统会将分配的uid和Package name生成一条记录写入packages.list中，本方案通过在内核中读取Package name并通过packages.list获得uid，作为网络管控模块对应用进行管控的判断依据. 在本测试环境下，系统中的uid和Package name的记录如图5所示.

5.3 Wi-Fi下网络白名单测试

为了对Wi-Fi下手机联网情况进行分析，本测试用计算机作为手机联网的代理，在计算机端对来自手机

表4 网络白名单测试环境

类别	配置信息
手机型号	Samsung Galaxy S10
手机存储	128 GB
手机内存	8 GB
手机cpu	Exynos 9820
Android版本	Android 10
内核编译环境	Ubuntu 16.04
内核Linux版本	4.19.87
基带版本	G973FXXU7CTF1

```
com.samsung.android.provider.filterprovider 1000 0
/data/user/0/com.samsung.android.provider.filterprovider
platform:privapp:targetSdkVersion=24
2001,1065,5050,3002,1023,3003,3001,3007,1002,3010,3011,1024,1007 0
501100000
com.sec.android.app.DataCreate 10221 0
/data/user/0/com.sec.android.app.DataCreate platform:targetSdkVersio
3002,1023,3003 0 1
com.android.cts.priv.ctsshim 10064 0
/data/user/0/com.android.cts.priv.ctsshim
untrusted:privapp:targetSdkVersion=28 none 0 28
com.sec.android.widgetapp.samsungapps 10079 0
/data/user/0/com.sec.android.widgetapp.samsungapps
samsung:privapp:targetSdkVersion=24 3003 0 171302400
com.samsung.android.smartswitchassistant 1000 0
/data/user/0/com.samsung.android.smartswitchassistant
platform:privapp:targetSdkVersion=24
2001,1065,5050,3002,1023,3003,3001,3007,1002,3010,3011,1024,1007 0
110600000
com.sec.vsim.ericssonnsds.webapp 10105 0
```

图5 packages.list中uid和Package name的对应实例

的网络请求使用wireShark进行抓包. wireShark是一个功能强大的网络数据包分析软件，可以截获各种网络数据包，并显示数据包的详细信息.

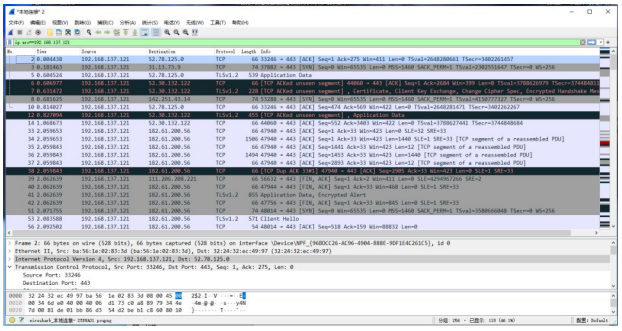
计算机开启热点，手机连接后查询到对应的ip，如图6(*a*). 在wireShark中过滤source等于手机ip的数据包并进行显示，如图6(*b*). 在没有开启网络管控时，计算机端可以一直抓到手机发送的各种网络数据包.

通过以下四个步骤对网络白名单进行测试：

已连接的设备:　　1台(共 8 台)

设备名称	IP 地址	物理地址(MAC)
Galaxy-S10	192.168.137.121	ba:56:1e:02:83:3d

(*a*) 手机联网IP



(*b*) wireShark实时抓包

图6　WireShark抓取手机网络数据包

(1)在手机端开启网络白名单管控功能,并开启80端口管控功能,结果如图7(*a*),所有的TCP网络数据包都被禁止,仅有DNS(Domain Name System)解析相关包可见.

(2)将百度应用加入到白名单后,除了有发送到本地192.168.137.1的数据包外,其他数据包的目的ip均为百度ip,结果如图7(*b*). 此时在手机端,除了百度应用可以正常联网外,其他的应用均不能联网.

(3)关闭80端口管控功能,结果如图7(*c*). 抓包结果中出现了一些端口号为80的数据包.

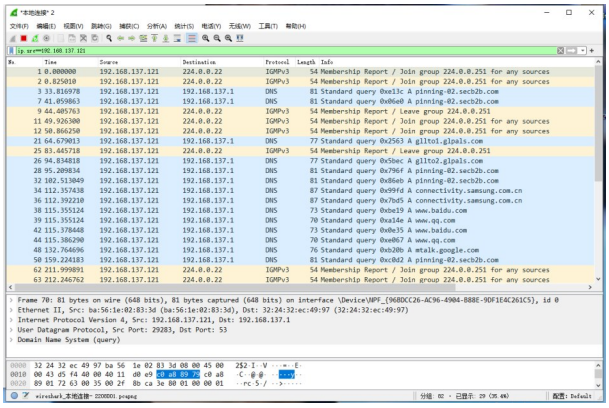
(4)关闭网络白名单管控功能和80端口管控功能,结果如图7(*d*),不在白名单中的应用也开始发送数据包.

重复以上步骤,在第(2)步将不同的应用加入白名单,测试结果表明,Wi-Fi下只有在白名单里的应用可以联网.

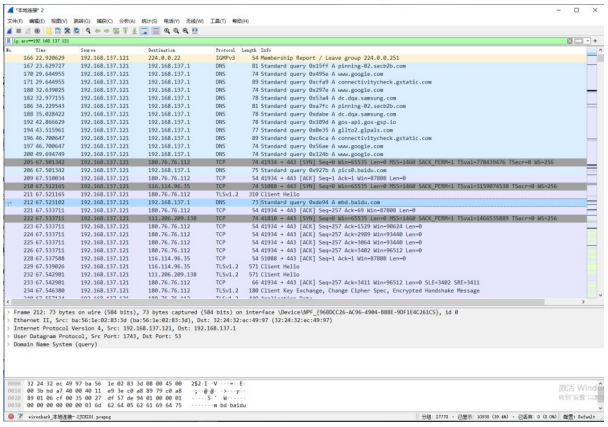
5.4　移动数据下网络白名单测试

手机使用移动数据联网时通过Modem直接将数据发往基站,因此需要在Android手机中实现对网络数据包的截获. tcpdump是一个用于截取网络数据包,并输出分组内容的工具. 它有着灵活的截取策略,无需root,操作简单方便,且可以通过命令在后台运行,是在Linux下排查网络问题的首选工具. 本测试使用tcpdump和wireShark相结合的方式,在手机中安装并运行tcpdump,抓取网络数据包并保存到存储设备中,在计算机端用wireShark打开并对数据包进行分析.

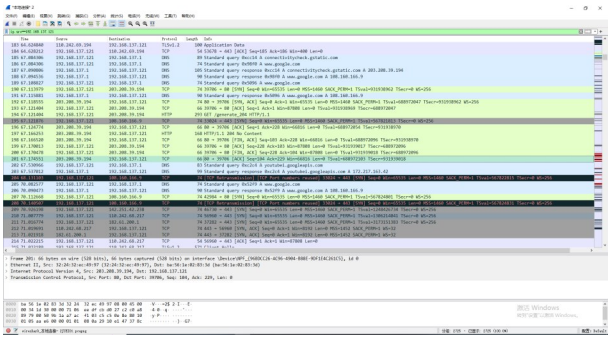
使用tcpdump -i any -p -s 0 -w sim-noAlarm.log等格式的命令运行tcpdump,此时tcpdump会在后台运行,不



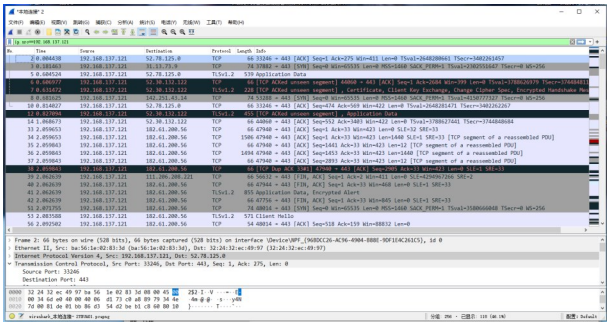
(*a*) 网络全部管控



(*b*) 百度应用加入白名单



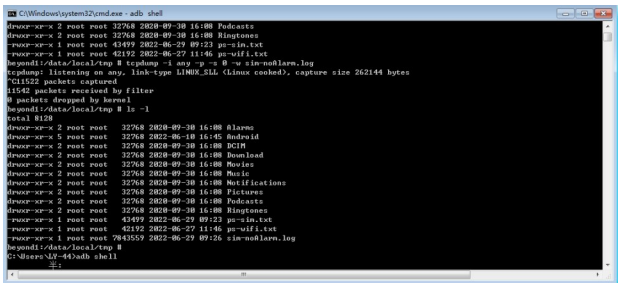
(*c*) 关闭80端口管控



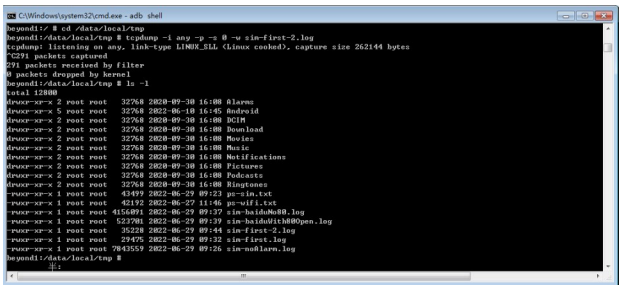
(*d*) 关闭网络管控

图7　Wi-Fi下网络白名单功能测试

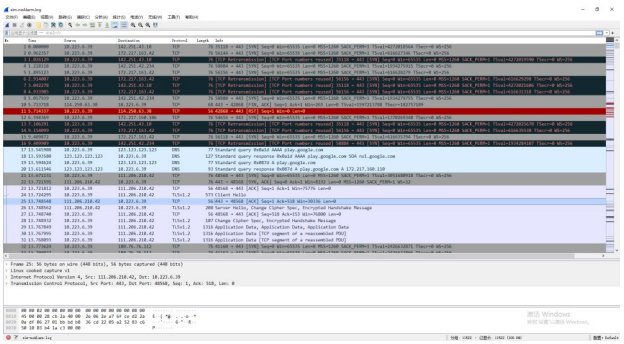
断抓取手机发送的数据包. 如图8(*a*). 抓取完毕后在手机内部存储生成sim-noAlarm. log等命名的文件,如图8(*b*). 将这些log文件拷贝到计算机中,通过wireShark打开,即可看到此次抓取的网络数据包情况,如图8(*c*).



(*a*) 运行tcpdump



(*b*) 数据包保存文件tcpdump.log



(*c*) wireShark读取tcpdump.log

图8 tcpdump抓取手机网络数据包

测试流程和5.3节的4个步骤相同,分别测试打开网络白名单管控功能和80端口管控功能、加入一个应用到白名单、关闭80端口管控功能、关闭网络白名单管控功能四种情况下手机的发包情况. 测试结果和图7所展示的结果相同,此处不再重复展示. 测试结果表明,移动数据下只有在网络白名单上的应用可以发送网络数据包.

5.5 网络管控环境下系统性能评估

为了对运行了网络白名单系统的性能进行量化评估,本节采用鲁大师应用进行性能评测. 鲁大师是一款权威评测软件,可以对系统的性能总分、CPU(Central Processing Unit)、GPU(Graphics Processing Unit)、RAM(Random Access Memory)、ROM(Read-Only Memory)等进行评测,如图9为一次评测结果.

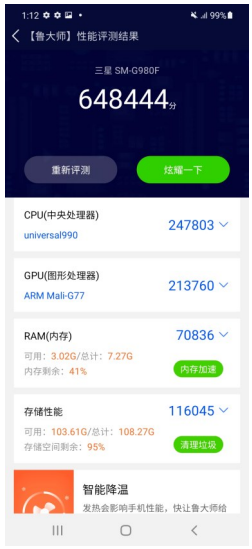


图9 系统性能评测方法

为了更准确地评测系统性能,本测试连续十天分别对运行了白名单的系统和未运行白名单的系统进行了评测,期间手机正常使用,十次评测总分如图10.

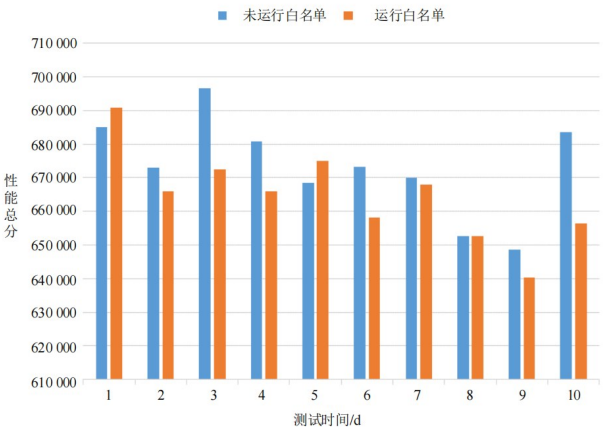


图10 白名单十次性能评测总分统计

测试结果表明,相对于原系统,运行了白名单的系统性能总体上下降1.28%.

5.6 网络管控环境下应用性能评估

本测试选择了十款常用的应用,测试网络管控对其是否有效,同时对每款应用还分别测试了十次禁止使用网络前后的启动时间,以评估应用性能损失情况. 测试结果如表5所示. 测试结果表明,网络白名单对应用的网络权限管控成功率为100%. 同时,性能损耗最

多的应用启动时间增加了162 ms,占原启动时间的33.1%,性能损耗最少的应用启动时间增加了9 ms,占原启动时间的3.6%.

表5　网络管控环境下应用性能评估

应用	网络管控是否有效	禁止前平均启动时间/ms	禁止后平均启动时间/ms
百度	有效	223	379
UC浏览器	有效	127	224
抖音	有效	382	432
bilibili	有效	284	360
高德地图	有效	210	340
支付宝	有效	490	652
微信	有效	980	1 110
计算器	有效	127	145
番茄闹钟	有效	243	284
天猫精灵	有效	248	257

6　结论

本文在Android系统上,针对恶意应用利用互联网泄露用户数据的行为,研究了一套网络白名单系统. 普通应用无法在应用层对其他应用进行网络管控,本方案结合内核的设备驱动实现了该功能. 同时本文构建了管控应用和内核的专用通信通道,避免指令被其他应用使用. 用户可以在本系统的管控应用中监控其他应用的网络使用情况,并选择信任的应用加入白名单中,实现对应用的网络管控.

经过实验验证,本文提出的网络白名单方案可以有效防止恶意应用利用互联网泄露用户隐私,对应用的网络管控成功率达到了100%. 整套系统使用方便,运行稳定,被管控应用启动时间最大增加33.1%,最小增加3.6%.

本文所述方案是一个通用的方案,其基于Android的系统软件层面完成,具备硬件无关性,方案通过修改Linux内核代码实现权限控制来完成应用管控功能,对Modem驱动的网络管控操作在modem_io_device.c中完成,对wifi驱动的网络管控操作在dhd_linux.c中完成. 相关修改方法在多个版本的Android系统中均可适用,可应用于Android的最新版本及相关机型. 在下一步工作中,网络白名单功能可以对应用发送的数据进行鉴别,分别对敏感数据和普通数据进行管控,从而保证一些应用的正常联网行为.

参考文献

[1] 贡知洲, 路昭亮. Android发展的分析与研究[J]. 价值工程, 2013, 32(2): 185-186
GONG Z Z, LU Z L. Analysis and research of the android [J]. Value Engineering, 2013, 32(2): 185-186 (in Chinese)

[2] StatCounter. Mobile operating system market share worldwide[EB/OL]. (2021-08-31) [2022-04-27]. https://gs. statcounter.com/os-market-share/mobile/worldwide.

[3] 李鹏伟, 姜宇谦, 薛飞扬, 等. 一种基于深度学习的强对抗性Android恶意代码检测方法[J]. 电子学报, 2020, 48(8): 1502-1508.
LI P W, JIANG Y Q, XUE F Y, et al. A robust approach for android malware detection based on deep learning[J]. Acta Electronica Sinica, 2020, 48(8): 1502-1508. (in Chinese)

[4] 董超, 杨超, 马建峰, 等. Android系统中第三方登录漏洞与解决方案[J]. 计算机学报, 2016, 39(3): 582-594.
DONG C, YANG C, MA J F, et al. The vulnerabilities and solutions of third-party login services in android system[J]. Chinese Journal of Computers, 2016, 39(3): 582-594. (in Chinese)

[5] XUE Y, ZHANG X S, YU X, et al. Isolating host environment by booting android from OTG devices[J]. Chinese Journal of Electronics, 2018, 27(3): 617-624.

[6] ZIMPERIUM. Global-mobile-threat-report[EB/OL]. (2022-03-21) [2022-04-28]. https://www. zimperium. com/global-mobile-threat-report/.

[7] CUI H L, SHAO S, NIU S Z, et al. Container-based privacy preserving scheme for android applications[J]. Chinese Journal of Electronics, 2020, 29(4): 731-737.

[8] HU J L, LIANG J J, KUANG Y Z , et al. A user similarity-based Top-*N* recommendation approach for mobile in-application advertising[J]. Expert Systems with Applications, 2018, 111: 51-60.

[9] SARACINO A, SGANDURRA D, DINI G, et al. MADAM: Effective and efficient behavior-based android malware detection and prevention[J]. IEEE Transactions on Dependable and Secure Computing, 2018, 15(1): 83-97.

[10] ZHANG W, SU N N, NIU S Z, et al. A novel hotfix scheme for system vulnerability based on the android application layer[J]. Chinese Journal of Electronics, 2019, 28(2): 408-415.

[11] Google. Android 架构 [EB/OL]. (2020-03-21) [2022-04-15]. https://source.android.google.cn/devices/architecture?hl=zh-cn.

[12] 卿斯汉. Android安全研究进展[J]. 软件学报, 2016, 27(1): 45-71

QING S H. Research progress on android security[J]. Journal of Software, 2016, 27(1): 45-71 (in Chinese)

[13] 庞振海. 基于Android平台的双文件系统的设计与实现[D]. 北京: 北京邮电大学.
PANG Z H. Design and Implementation of Dual File System Based on Android Platform[D]. Beijing: Beijing University of Posts and Telecommunications. (in Chinese)

[14] Sinha N . Android-Based Driver Alert System for Accident Avoidance[M]. Delhi: Springer,2021.

[15] 路子聪, 徐开勇, 郭松, 等. 基于ARM虚拟化扩展的Android内核动态度量方法[J]. 计算机应用, 2018, 38(9): 2644-2649.
LU Z C, XU K Y, GUO S, et al. Dynamic measurement of Android kernel based on ARM virtualization extension [J]. Journal of Computer Applications, 2018, 38(9): 2644-2649. (in Chinese)

[16] 谢剑. 基于HTTP报文网络抓包软件的分析与设计[J]. 现代信息科技, 2020, 4(22): 20-22.
XIE J. Analysis and design of network packet capture software based on HTTP message[J]. Modern Informationn Technology, 2020, 4(22): 20-22. (in Chinese)

[17] MAJIDHA FATHIMA K M, SANTHIYAKUMARI N. A survey on network packet inspection and ARP poisoning using wireshark and ettercap[C]//2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). Piscataway: IEEE, 2021: 1136-1141.

[18] GUAN X J , MA Y Y , SHAO Z P, et al. Design and application of concurrent test scheme for heartbeat message of mobile terminal based on Tcpdump and LoadRunner [C]//2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC). Piscataway: IEEE, 2020: 232-235.

作者简介

杨易达 男,1998年出生于新疆维吾尔自治区叶城县,2020年于西安理工大学获得学士学位,现为西安理工大学在读硕士生,主要研究方向为网络与信息安全.
E-mail: yyd1998_xaut@qq.com

孙钦东 男,1975年出生于山东省莒南县,2005年于西安交通大学获得博士学位,现为西安理工大学计算机科学与工程学院教授,主要研究方向为网络与信息安全、物联网系统安全、人工智能安全等.
E-mail: qdongsun@xjtu.edu.cn

胡国星 男,1994年出生于安徽省淮南市,2022年于北京理工大学获得硕士学位,现为阿里巴巴后端开发工程师,主要研究方向为地图数据处理.
E-mail: 279973155@qq.com

李元章 男,1978年2月出生于江苏盐城,分别于2001、2004及2015年获得北京理工大学学士、硕士和博士学位,现为北京理工大学计算机学院副教授,主要研究方向为信息系统安全、人工智能安全等.
E-mail: popular@bit.edu.cn