Android 共谋攻击检测模型

杨宏宇，王在明
（中国民航大学计算机科学与技术学院，天津 300300）

摘 要：为了解决对 Android 共谋攻击检测效率差和准确率低的问题，提出基于组件通信的 Android 共谋攻击检测模型。首先，提取已知应用的特征生成特征向量集。其次，对权限特征向量集进行训练和分类，生成安全策略规则集。然后，根据组件和通信方式特征向量集生成组件通信有限状态机并优化安全策略规则集。最后，通过提取待测应用的特征向量集生成新状态机，与已优化安全策略规则集进行匹配检测共谋攻击。实验结果表明，所提检测模型具有较好的检测效率和较高的准确率。

关键词：Android 安全；共谋攻击；组件通信；安全策略规则集；有限状态机
中图分类号：TP309.7
文献标识码：A
doi: 10.11959/j.issn.1000-436x.2018095

Android collusion attack detection model

YANG Hongyu, WANG Zaiming
School of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China

Abstract: In order to solve the problem of poor efficiency and low accuracy of Android collusion detection, an Android collusion attack model based on component communication was proposed. Firstly, the feature vector set was extracted from the known applications and the feature vector set was generated. Secondly, the security policy rule set was generated through training and classifying the privilege feature set. Then, the component communication finite state machine according to the component and communication mode feature vector set was generated, and security policy rule set was optimized. Finally, a new state machine was generated by extracting the unknown application's feature vector set, and the optimized security policy rule set was matched to detect privilege collusion attacks. The experimental results show that the proposed model has better detective efficiency and higher accuracy.

Key words: Android security, collusion attack, component communication, security policy rule set, finite state machine

1 引言

随着移动智能终端的普及，Android 系统所面临的安全威胁越来越多。Android 共谋攻击是近年来新出现的一种安全威胁，具有攻击形式多样、隐蔽性强和攻击范围广等特点，已成为 Android 安全的研究热点课题[1]。

共谋攻击主要通过绕过 Android 安全机制（签名、沙盒等）的检查实施攻击[2-3]，相比其他类型的攻击，Android 共谋攻击隐蔽性更好、检测难度更高[4-6]。与检测恶意应用不同，检测共谋攻击不仅需要检测安全威胁的特征属性，还需检测应用间是否存在通信行为。由于现有大多数恶意应用检测技术未检测应用间的通信，因此不适用于检测共谋攻击[7]。为此，有研究提出如 FlowDroid[6]和 Amandroid[8]等基于污点分析技术的共谋攻击检测方法，通过设置污点追踪敏感信息流的泄露路径，达到检测共谋攻击的目的，但在效率方面表现不

收稿日期：2017-10-12；**修回日期**：2018-01-03
基金项目：国家科技重大专项基金资助项目（No.2012ZX03002002）；中国民航科技基金资助项目（No.MHRD201009, No.MHRD201205）
Foundation Items: The National Science and Technology Major Project (No.2012ZX03002002),The Science & Technology Project of CAAC (No.MHRD201009, No.MHRD201205)

佳。为改善上述检测方法的效率，APKCombiner 通过分析应用间的信息流提高检测共谋攻击的准确率，但该工具仅能检测出 2 个应用间的共谋攻击行为[9]。Schlegel 等[10]通过提取应用的公开和隐蔽信道信息来检测隐私数据泄露，但该方法单一且检测数据有限。Bartel 等[11]提出一种基于权限的共谋攻击检测方法，通过提取应用权限特征并静态分析函数调用检测共谋攻击，但检测准确率和效率不高。Sadeghi[12]提出一种检测应用间共谋攻击的 COVERT 工具，该工具虽然能够检测并显示组件间的通信，但在检测隐式组件方面误差大，导致检测攻击准确率低。

针对以上研究中检测共谋攻击效率差和准确率低的问题，本文提出基于组件通信的 Android 共谋攻击检测模型，目的是有效提高检测共谋攻击的效率和准确率。

2　检测模型

本文提出的 Android 共谋攻击检测模型结构如图 1 所示，该模型由训练阶段和检测阶段组成。

1) 训练阶段。提取已知样本所需的特征，生成特征向量集合；分析统计已知样本的特征关系图，保存组件通信与权限组合特征。

2) 检测阶段。提取未知样本指定的特征，生成特征向量集合；分析统计并生成未知样本的特征关系图，保存组件通信与权限组合特征；根据训练阶段的特征序列，在特征库中与检测阶段特征序列进行匹配；根据匹配结果输出未知样本的类型。

训练阶段负责对已知样本的训练和分类，并将样本的各种特征序列保存到数据库中；检测阶段则对未知样本进行检测，训练先于检测执行，并在数据库中将特征序列与检测阶段的特征序列进行匹配，输出匹配后的分类结果。

该模型的检测流程设计如下。

步骤 1　首先，使用 Androguard[13]反编译 APK，得到 AndroidManifest.xml 文件，再使用 python 编程，通过 xml.dom 模块和 I/O 模块对此文件解析，解析后将权限属性集、组件属性集、通信属性集保存到相应的文件中。然后，分别计算这 3 类属性集的数量并按使用频率从高到低排序后保存，得到新特征向量属性集。最后，将新特征向量属性集分成权限特征向量集、组件特征向量集及通信方式特征向量集并保存。

步骤 2　训练阶段。首先，使用改进贝叶斯的分类方法处理 2 类应用的权限特征向量集，对样本进行训练和分类后，将分类后的共谋应用权限组合集作为安全策略规则集保存到数据库中。然后，使用基于有限状态机的组件通信检测方法处理组件和通信方式特征向量集，生成应用间组件状态图。随后，编程实现状态机，检测应用通信状况，删除不存在通信行为的安全策略规则集。最后，将规则集与组件通信状态机保存到数据库中。

步骤 3　检测阶段。首先，使用改进贝叶斯的分类方法处理待测应用的权限特征向量集，提取出权限组合集并保存。然后，使用基于有限状态机的组件通信检测方法处理组件和通信方式特征向量集，提取组件通信状态机数据。最后，将权限组合集作为有限状态机的输入字符集生成新权限状态

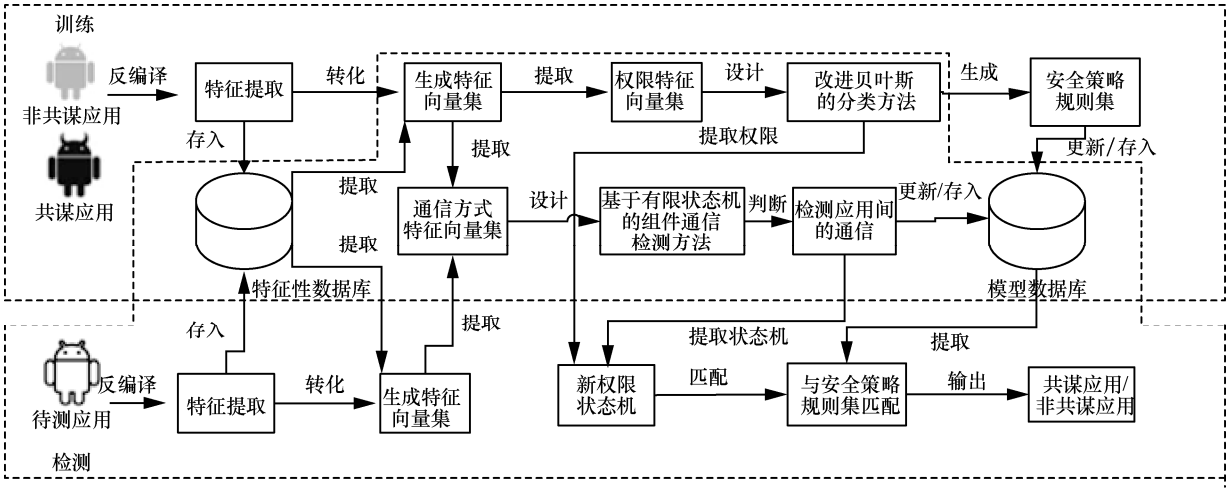

图 1　Android 共谋攻击检测模型结构

机，再与检测阶段保存的安全策略规则集进行匹配，输出匹配结果作为检测结果集。该阶段如算法 1 所示。

算法 1　检测阶段

输入　应用数据集 $\Lambda=\{app_1,app_2,app_3,\cdots,app_n\}$

输出　共谋应用对集合 $\Omega=\{pair_1,pair_2,pair_3,\cdots,pair_m\}$

if$|\Lambda| \geqslant 2$ then
　让 $\Theta=\Lambda$ 中所有可能的 *app* 组合;
　for each $pair_j$ in Θ do
　　改进贝叶斯的分类方法并计算 L_τ　;
　　if $L_\tau \geqslant$ *threshold* then
　　基于有限状态机的组件通信检测方法并计算 L_{com};
　　if L_{com} == 1 then
　　return ($pair_j$)
　　计算 L_c
　　end if
　end if
end for
end if

3　检测方法

在该模型中，设计了一种改进贝叶斯的分类方法和一种基于有限状态机的组件通信检测方法，上述 2 种方法为模型的核心方法，下面分别对 2 种方法进行说明。

本文中的符号说明如表 1 所示。

3.1　改进贝叶斯的分类方法

针对贝叶斯易出现无法比较、下溢出和分类差错率较高的问题，本文在贝叶斯算法的基础上对其进行了改进，以提高分类性能，改进过程如下。

首先，假设应用的权限为 W_i，共谋应用为 C，非共谋应用为 H。判断应用共谋与否，即判断权限集中是否包含权限 W_i，记为 $P(C|W_i)$，根据贝叶斯理论可得

$$P(C\mid W_i)=\frac{P(W_i\mid C)P(C)}{P(W_i\mid C)P(C)+P(W_i\mid H)P(H)} \quad (1)$$

其中，$P(C|W_i)$为出现权限 W_i 的应用是共谋应用的条件概率；$P(C)$为训练阶段应用中存在共谋应用的概率；$P(W_i|C)$为共谋应用中权限 W_i 出现的概率；$P(H)$为训练阶段应用中非共谋应用的概率；$P(W_i|H)$为非共谋应用中权限 W_i 出现的概率。

考虑到每个权限出现的独立性，计算 $P(C|\boldsymbol{W})$ 的联合概率 $P(C|\boldsymbol{W})$，$\boldsymbol{W}=\{W_1,W_2,W_3,\cdots,W_n\}$，如式(2)所示。

$$P=\frac{P_1P_2P_3\cdots P_n}{P_1P_2P_3\cdots P_n+(1-P_{1)}(1-P_2)(1-P_3)\cdots(1-P_n)} \quad (2)$$

表 1　符号说明

符号	说明	符号	说明	符号	说明
Λ	应用数据集	L_τ	分类阈值	W_i	应用的权限
Ω	共谋应用对集合	L_{com}	通信判定值	C	共谋应用
$P(C\|W_i)$	出现权限 W_i 的应用是共谋应用的条件概率	$P(C)$	应用中存在共谋应用的概率	H	非共谋应用
$P(W_i\|C)$	共谋应用中权限 W_i 出现的概率	$P(H)$	应用中非共谋应用的概率	$P(W_i\|H)$	非共谋应用中权限 W_i 出现的概率
G	有限状态机	Q	应用所有组件状态的非空有限集合	Σ	输入字符表（符号的非空有限集合）
q_0	组件某一初始状态	δ	状态转移函数	F	接受（最终）状态集合
A	应用	α	Activities 组件的集合	β	A 中 Services 组件的集合
γ	A 中 Broadcast Receivers 组件集合	ξ	A 中与 Intent 相关的 API 调用	$\zeta(\xi)$	ξ 中所有操作字符串的集合
V	图的点集合	$S(\zeta)$	ξ 中发送 Intent 传递信息的组件集	$T(\xi)$	接收 Intent 处理信息的组件集
E	图的边集合	$G(V,E)$	应用的图表示	$G^u=(V,E)$	应用图的融合表示
ξ_i	Intent 的 API 调用	$S(\xi_k)$	ξ_k 中发送和接收 Intent 的组件集	$G^d=(V^d,E^d)$	应用融合图优化后表示
E^d	融合图的边	V^d	融合图的点	M	状态机
TP	真正	FP	误报	TN	真负
FN	漏报	TPR	真正率	FPR	误报率
ACC	准确率	ERR	差错率		

其中，$P=P(C|\boldsymbol{W})$为出现权限 $\boldsymbol{W}=\{W_1,W_2,W_3,\cdots,W_n\}$ 的应用为共谋应用的条件概率；$P_i=P(C|W_i)$为出现权限 W_i 的应用是共谋应用的条件概率。

其次，鉴于贝叶斯计算概率时存在无法比较和下溢出的问题[14]，当权限不存在，即 $W_i=0$ 时，$P_i=0$，$P=0$，则无法比较；当 P_i 较小时，计算 P 会造成下溢出问题。针对上述 2 个问题，本文分别提出 2 种解决方案。

判断样本中权限特征向量 $\boldsymbol{W}=\{W_1,W_2,W_3,\cdots,W_n\}$ 中的权限 $W_i=0$ 是否为 0，如果为 0，则计算 $P(W_i|C)$ 和 $P(W_i|H)$时，将所有权限初始化出现的次数均记为 1，再进行计算得到 P_i。

当 $P_i<0.000\ 1$ 时，若要计算 $P(W_i|C)$和 $P(W_i|H)$，取 $P(C|W_i)$的对数 $\ln\{(P(C|W_i))^{-1}\}$，将最终结果变为计算 $P(W_1|C)P(W_2|C)\cdots P(W_n|C)P(C)$ 和 $P(W_1|H)P(W_2|H)\cdots P(W_n|H)P(H)$的大小。

最后，为降低分类差错率，在计算联合概率 $P(C|W)$时引入一个变量调整因子 θ，其作用是调整权限列表中某一权限组合共谋的联合概率。θ 值越小表示分类越准确，θ 的选取范围为[0.01, 0.1]。通过 Adaboost 算法[15]对调整因子 θ 迭代多次后取最佳值。改进贝叶斯的分类方法如算法 2 所示。

算法 2 改进贝叶斯的分类方法

输入 权限特征向量集

输出 权限组合集和 $P(C|\boldsymbol{W})$等

1) *count* =*n* /*设定 Adaboost 循环次数 *count*/

2) *sample*=2 000 /*选择 2 000 个样本*/

3) $|\theta|=|S|$ /*θ 初始化为和权限列表大小相等的向量*/

4) for *t* =0 to *count* /*迭代*/

5) *inf* =0.5 % /*设定最小分类差错率为 *inf*/

6) *value*=*classfy*(*sample*) /*对于每一个样分类*/

7) if(*value* == error) /*如果分类出错*/

8) if(*alpha*(*pc*)>*alpha*(*ph*)) /*计算出错的程度即比较 $P(C)$和 $P(H)$的相差 *alpha*/

9) $\theta[S]$ = np.abs($\theta[S]-\frac{\text{np.exp}(alpha)}{DS[S]}$) /*若样本原本是 C，错分成 H*/

10) else /*若样本原本是 H，错分成 C*/

11) $\theta[S]=\theta[S]+\frac{\text{np.exp}(alpha)}{\theta[S]}$

12) *inf*=*compute*(*S*) /*计算差错率*/

13) *Save*(*inf*,$P(W_i|C)$,$P(W_i|H)$ /*保存差错率和此时的权限集、$P(W_i|C)$和 $P(W_i|H)$、θ 等信息*/

通过执行改进贝叶斯的分类方法，得到权限组合集和联合概率 $P(C|\boldsymbol{W})$等信息。为方便计算，设定 $L_\tau(S)=\ln\{(P(C|\boldsymbol{W}))^{-1}\}$，取值为[0,1]，当 $L_\tau(S)>0.5$ 时，将权限组合集保存为安全策略规则集。

3.2 基于有限状态机的组件通信检测方法

为检测应用间的通信情况，本文设计一种基于有限状态机的组件通信检测方法，用 $G=<Q,\Sigma,\delta,q_0,F>$表示有限状态机[16]，其中，$Q$ 表示应用所有组件状态的非空有限集合；Σ 表示输入字符表（符号的非空有限集合）；q_0 是组件的某一初始状态，它是 Q 的元素；δ 是状态转移函数，即 δ：$Q\times\Sigma\to Q$；F 是接受（最终）状态的集合。

以此为基础，基于有限状态机的组件通信检测方法的设计步骤如下。

步骤 1 将应用转化为图。

若用 A 表示一个应用，α 表示 A 中 Activities 组件的集合，β 表示 A 中 Services 组件的集合，γ 表示 A 中 Broadcast Receivers 组件的集合，ξ 表示 A 中与 Intent 相关的 API 调用，$\zeta(\xi)$表示 ξ 中所有操作字符串的集合。

定义点集合 V 为

$$V=\alpha\cup\beta\cup\gamma\cup\zeta(\zeta) \tag{3}$$

其中，V 代表应用的点集合。

用 $S(\xi)$表示 ξ 中发送 Intent 传递信息的组件集，$S(\xi)=\{(x,i)|x\in\{\alpha\cup\beta\cup\gamma\},x$ 发送 Intent $i\in\xi\}$，$T(\xi)$表示接收 Intent 处理信息的组件集，$T(\xi)=\{(x,i)|x\in\{\alpha\cup\beta\cup\gamma\},x$ 接收 Intent $i\in\xi\}$。

定义边集 E 为

$$\begin{aligned}E=\{(x,y)\,|\,x\in S(\xi),y\in T(\xi)\cup\\ \{(x,y)\,|\,x\in S(\xi),y\in\varsigma(\xi)\cup\\ \{(x,y)\,|\,x\in T(\xi),y\in\varsigma(\xi)\end{aligned} \tag{4}$$

用图 $G(V,E)$表示应用 A，图 2 为 4 个 Android 典型应用的状态图。

步骤 2 图的融合。

为处理应用间的隐式 Intent 调用问题，将应用图 $G(V,E)$融合。设有 n 个应用，由步骤 1 将应用转

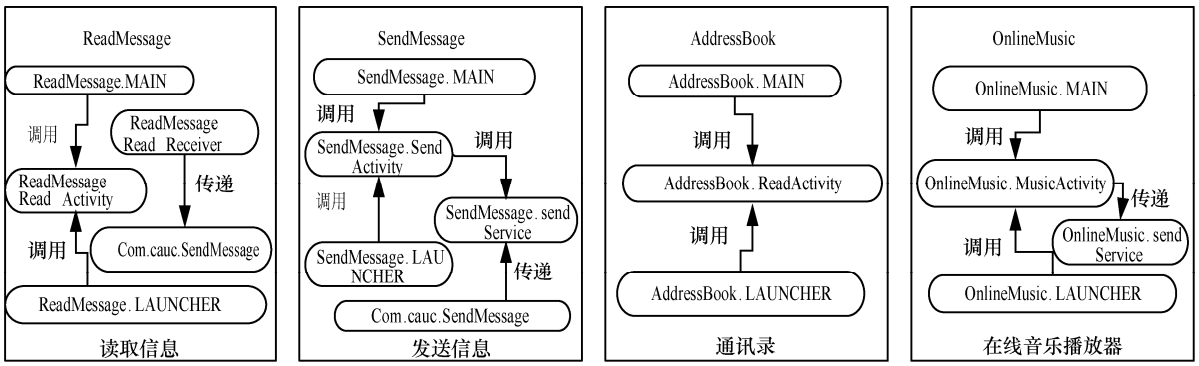

图 2　4 个典型应用的状态图

化为对应的状态图，所有图的融合表示为 G^u

$$G^u=\bigcup_{i=1}^{n}G_i, G_i=(V_i,E_i), \forall i=1,\cdots,n \tag{5}$$

$G^u=(V,\ E)$，其中，$V=V_1\cup V_2\cup V_3\cdots\cup V_n$，$E=E_1\cup E_2\cup E_3\cdots\cup E_n\cup E'$，$E'$表示应用间发送和接收隐式 Intent 信息的边。任意 A_i 和 A_j（$i\neq j\in\{1,\cdots,n\}$），都有对应的图 G_i 和 G_j，用 ξ_i 或 ξ_j 表示 A_i 或 A_j 中与 Intent 有关的 API 调用。令 $k=i,j$ ($i\neq j\in\{1,\cdots,n\}$)，则 $S(\xi_k)$和 $T(\xi_k)$分别表示 ξ_k 中发送和接收 Intent 的组件集。因此，E'可表示为：$E'=\{(x,y)\mid \exists i,j\in\{1,\cdots,n\}$ s.t. $x\in S(\xi_k), y\in T(\xi_k), i\neq j\}$。

将 4 个 Android 典型应用的状态图融合后的结果如图 3 所示。

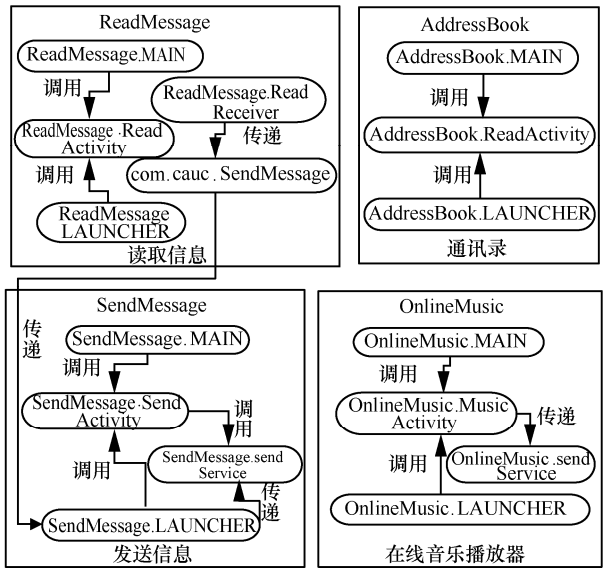

图 3　经融合后的 4 个典型应用的状态图

步骤 3　融合图的优化。

为保证融合图的计算效率和算法的执行速度，需删除动作字符串的节点（即在 G^u 中删除 $\zeta(\xi)$）。删除融合图节点的规则设计如下。

1) 若某个组件（图节点）没有进来的边，只有出去的边，则删掉此节点。

2) 若某个组件(图节点)有来自其他组件的边，则保留组件（图节点）的边并删掉对应的动作字符串节点。

融合图优化后，G^u 记为 $G^d=(V^d,\ E^d)$，边记为 $E^d=E\cup\{S(\xi)\to T(\xi)\mid S(\xi)\to x\to T(\xi), x\in\zeta\}\setminus\{\{(x,y)\mid x\in S(\xi), y\in\zeta(\xi)\}\cup\{(x,y)\mid x\in T(\xi), y\in\zeta(\xi)\}\}$，点记为 $V^d=V-\zeta(\xi)$。4 个典型应用的融合图优化后的结果如图 4 所示。

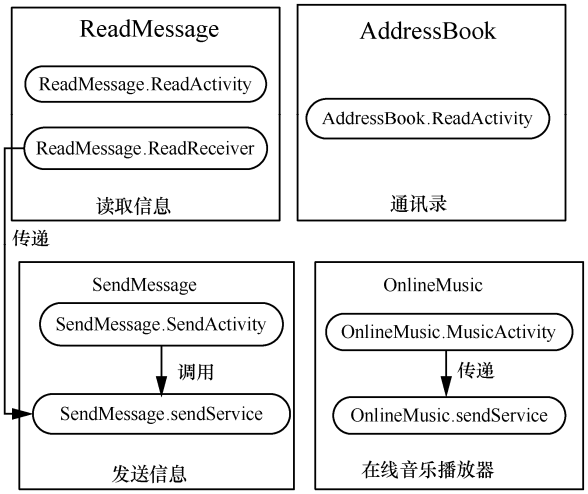

图 4　4 个典型应用融合图优化后的结果

步骤 4　优化图的状态机设计。

将优化图 G^d 设计成状态机 M 中对应的元素，G^d 节点作为状态机中接受状态集 F。初始状态为 q_0，权限集 $Q=F\cup\{q_0\}$，状态转移函数 δ 包含从 $q_0\to F$ 的转换，且能够访问 F 中的元素和 G^d 中的所有边。4 个典型应用的状态机如图 5 所示。

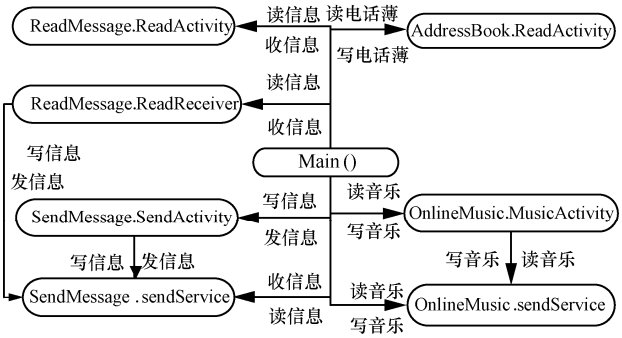

图 5 4 个典型应用的状态机

图 5 中，Main()是状态机的初始状态，根据组件间的关系便可定义状态转换函数 δ，最终状态为 ReadMessage.ReadActivity、AddressBoost.ReadAcitvty、SeedMessage.sendService、Online Music.seadService。算法 3 为计算 δ 的算法，算法 4 为计算基于有限状态机的组件通信检测的算法。

算法 3 计算 $\delta(Q,\Sigma)$

1) $m = length[Q]$

2) for q=0 to m

3) for each character $a \in \Sigma$ /*Σ 为训练得到的输入字符*/

4) k=min(m+1,q+2) /*危险权限状态值取最小*/

5) repeat

6) k=k–1

7) until $P_k > P_q a$

8) $\delta(q,a) = k$ /*加进权限 a 后的状态值为 k*/

9) return δ

算法 4 计算基于有限状态机的组件通信检测 (Q,δ,F)

1) $n = length[Q]$

2) q_0=0 /初始状态 q_0/

3) for j=1 to n

4) do $q_0 = \delta(q_0, Q[j])$ /*δ 函数是一张图表*/

5) if $q_0 = F$ /*F 为接受状态*/

6) then print “组件间存在通信状态转换” j~F

7) $L_{com}(S)$=1

8) else

9) then print “不存在通信状态转换” j~F

10) $L_{com}(S)$=0

经过基于有限状态机的组件通信检测方法处理，若应用间存在通信，则 $L_{com}(S)$=1，保存应用组件通信相关的数据，以便将优化安全策略规则集和将权限特征向量集作为输入字符集生成新权限状态机。

4 实验与结果

4.1 环境及样本设置

分别对本文提出的检测模型进行可行性验证、分类效果验证、共谋攻击与权限的关联性验证、不同方法的对比检测等实验，实验环境和样本设置如下。

实验环境：Dell PC 机；处理器 Interl(R) Core(TM) i7-3210 CPU；内存 4 GB；Pycharm2016 编译器；Matlab R2014a；

样本：训练样本为 1 000 个非共谋应用[17]和 1 000 个共谋应用[18]，非共谋应用从 Google Play 以及国内应用市场中下载，共谋应用从 VirusShare[18]中获得。检测样本为 300 个共谋应用和 200 个非共谋应用。为保证样本的纯度，所有样本均在 VirusTotal[19]中检测后使用。

为衡量分类器的分类效果，定义了以下检测指标。

指标 1 真正（*TP*）：实际为共谋应用，被检测为共谋应用。

指标 2 误报（*FP*）：实际为非共谋应用，被检测为共谋应用。

指标 3 真负（*TN*）：实际为非共谋应用，被检测为非共谋应用；

指标 4 漏报（*FN*）：实际为共谋应用，被检测为非共谋应用。

进一步定义以下指标。

指标 5 真正率：$TPR=\dfrac{TP}{TP+FN}$，表示检测出的共谋应用占实际共谋应用总数的比例。

指标 6 误报率：$FPR=\dfrac{FP}{FP+TN}$，表示检测出的共谋应用占实际非共谋应用总数的比例。

指标 7 准确率：$ACC=\dfrac{TP+TN}{TP+TN+FP+FN}$，用来衡量总体分类检测精度，该值越高则准确率越好。

指标 8 差错率：$ERR=\dfrac{FP+FN}{TP+TN+FP+FN}$，

用来衡量总体分类检测的差错，该值越小分类效果越好。

实验中采用分层 10 折交叉法来提高模型的泛化能力，选取 2 000 个训练样本作为训练集，选取 500 个检测样本作为测试集。迭代 10 次，每次随机选取测试集计算分类器的 *TPR*、*FPR*、*ACC*、*ERR*，然后取平均值。

4.2　可行性验证

首先，使用 Stateflow[20]工具对有限状态机进行仿真，仿真的设计如下。

1) 根据应用所求状态的数量、状态转移的条件（权限组合形式）和各状态输出信号的赋值，画出状态转移图。

2) 按照状态转移图编写状态机的设计程序。

3) 利用 Stateflow 工具对状态机的功能进行仿真验证。

4) 匹配成功则将权限所属应用标为共谋应用，否则为非共谋应用。

以 4 个共谋应用生成的状态转移图（如图 6 所示）为例。由图 6 可知，应用的组件间存在共谋攻击，此时安全策略规则集为权限 WRITE_EXTERNAL_STORAGE、SEND_SMS、INTERNET 和 READ_CONTACT 的组合，仿真结果输出这些应用为共谋应用。仿真结果输出为共谋应用的标记为 1，否则为 0。

然后，在 Pycharm 和 Matlab 中编程实现模型检测流程，对样本进行处理和检测分类。通过将 500 个检测样本导入检测模型，计算标记为 1 的应用数量，再通过 python 编写脚本对检测过程可视化，将检测的过程转换为标记 0 或 1 的过程，观察检测过程的变化，以验证模型的可行性。以 1 800 s 为间隔按时序截取 8 张检测过程图并记录下当时对样本的检测结果。图 7 为检测过程，黑色标记表示为 1 的样本，初始值根据样本总量全标为 1，最终检测结果为发现 286 个共谋应用。

由图 7 可知，从时序 *a* 到 *h* 标记为 1 的样本由 481 个最终稳定为 286 个，说明检测完成且共谋应用为 286 个，此时得到各个指标分别为

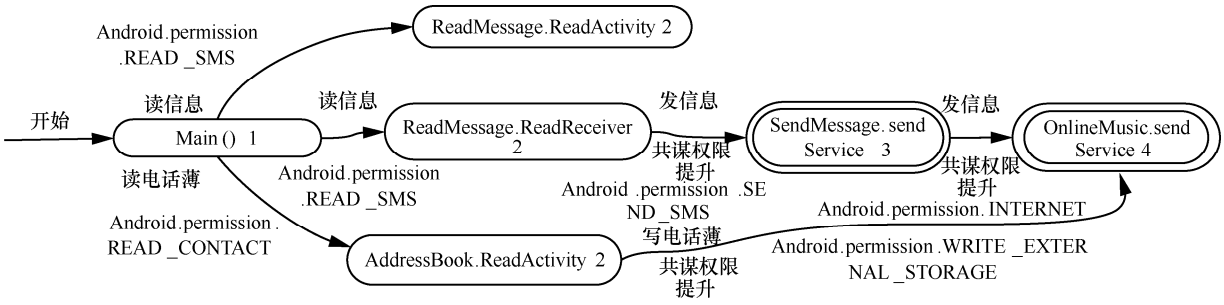

图 6　4 个共谋应用生成的新状态转移图

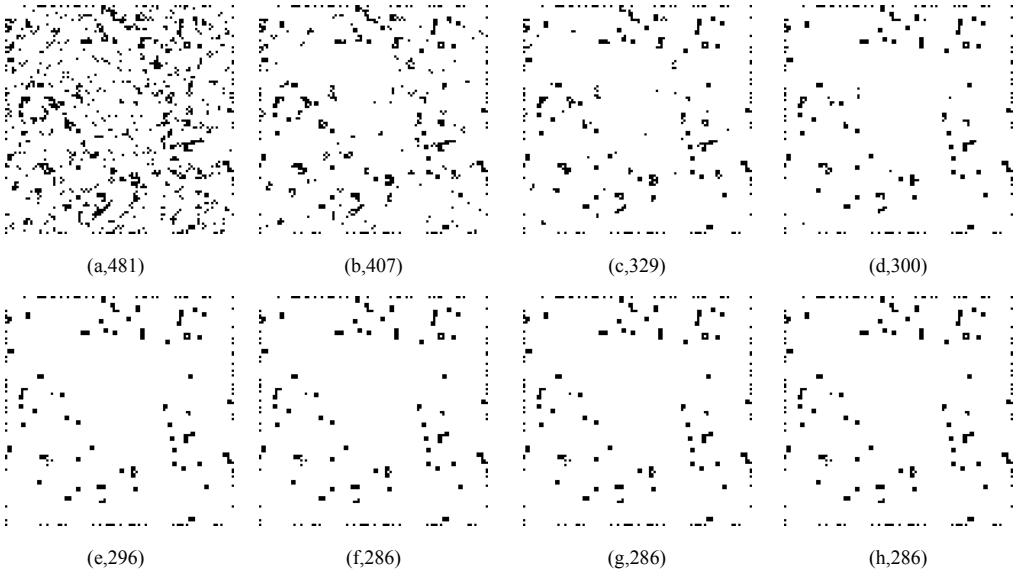

图 7　检测过程

TPR=93.33%，*FPR*=14.66%，*ACC*=92.45%，*ERR*=12.37%。实验表明本文模型检测共谋应用的真正率和准确率较高，且误报率较低。

4.3　分类效果验证

为验证模型的分类检测效果，将检测样本输入检测模型，经处理得到分类效果，如图 8 所示。由图 8 可知，样本中共谋应用与非共谋应用在 L_τ 约为 0.5 处（图 8 中黑色线）分离，此值上方样本均为共谋应用，下方样本均为非公谋应用，这一结果表明本文模型的分类效果较好且分类的阈值稳定。

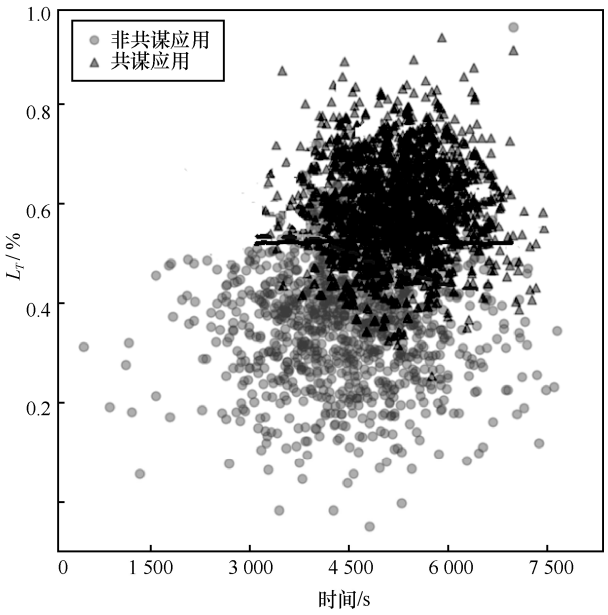

图 8　模型的分类效果

4.4　共谋攻击与权限的关联性验证

为验证共谋攻击与权限之间的关联性，统计出 2 类样本中 10 个最常用权限占各自样本使用权限总量的概率，并进行排名，结果如表 2 所示。由表 2 可知，在权限相同的情况下，共谋应用概率大于非共谋应用概率，说明共谋应用申请的权限数量更多，表明应用申请的权限数量越多，其存在共谋风险的可能性就越大。

为进一步验证应用申请权限数量与共谋攻击之间存在的关联性，对共谋样本的每个应用中 10 个最常用权限的数量与共谋攻击之间的关系进行统计，统计结果如图 9 所示。由图 9 可知，10 个权限中随着申请数量的增加，产生共谋风险的可能性增大。依据是权限数量越多，可能的共谋权限组合也就越多。

表 2　共谋与非共谋应用权限概率排名对比

权限名称	共谋应用 概率(排名)	非共谋应用 概率(排名)
INTERNET	95(1)	84(1)
ACCESS_NETWORK_STATE	92(2)	78(2)
READ_PHONE_STATE	89(3)	82(5)
WRITE_EXTERNAL_STORAGE	84(4)	65(3)
ACCESS_WIFI_STATE	80(5)	42(6)
WAKE_LOCK	75(6)	58(4)
ACCESS_COARSE_LOCATION	73(7)	30(10)
ACCESS_FINE_LOCATION	70(8)	36(9)
RECEIVE_BOOT_COMPLETED	63 (9)	46(7)
VIBRATE	60(10)	39(8)

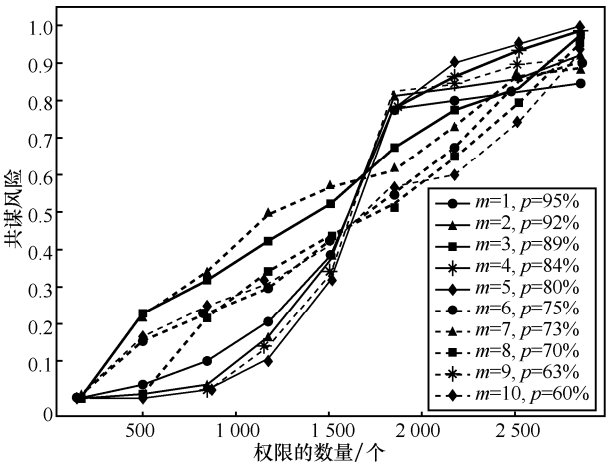

图 9　共谋风险与权限数量的关系

4.5　检测对比实验

为验证本文模型的检测准确率和检测效率，使用本文模型、当前的检测模型和工具，在相同条件下对样本进行检测并对比。

首先，编程实现 IC3 工具[21]和本文模型，然后对共谋测试套件 DroidBench[22]和共谋样本 APP 进行检测，其中，DroidBench 包含 29 个共谋测试应用，共谋样本 APP 是从样本中随机选取的 100 个 APP，分别获取检测失败的 APP 数、Intent 传递个数、耗时，并对上述检测数据进行统计，统计结果如表 3 所示。由表 3 可知，本文模型与 IC3 相比，在检测的准确率、Intent 传递数目、耗时方面均具有一定优势。

为进一步比较本文模型的检测性能，从共谋样本中随机抽取 20 个应用，分别使用 DroidBench 和共谋测试套件 ICC-Bench[23]，对 Amandroid 工具[8]、COVERT 工具[13]、DroidSafe 模型[24]和本文模型进行共谋攻击检测实验。使用上述工具和模型分别测试 10 次，测试结果如表 4 所示。由表 4 可知，在

DroidBench 和 ICC-Bench 测试环境下，与 Amandroid、COVERT 和 DroidSafe 相比，本文模型在检测共谋攻击方面的准确率更高。

表 3　IC3 工具和本文模型的对比分析

检测工具	DroidBench			共谋样本 APP		
	失败/个	Intent 传递/个	耗时/s	失败/个	Intent 传递/个	耗时/s
IC3 工具	0	850	1 600	20	3 145	6 300
本文模型	0	1 030	1 450	5	4 029	6 090

表 4　不同工具不同平台下相同样本检测效果对比

检测环境	样本	Amandroid	COVERT	DroidSafe	本文模型
DroidBench	1	N/A	Y	Y	N/A
	2	Y	N/A	Y	Y
	3	N	N/A	N	Y
	4	Y	N/A	N	Y
	5	Y	N/A	Y	Y
	6	Y	N/A	N	N/A
	7	N	N/A	N	Y
	8	N/A	N/A	N	Y
	9	N/A	N/A	Y	Y
	10	Y	Y	Y	Y
ICC-Bench	11	Y	N/A	N	Y
	12	N/A	N/A	Y	Y
	13	Y	N/A	Y	Y
	14	N	N/A	Y	Y
	15	Y	N/A	Y	Y
	16	Y	N/A	Y	Y
	17	N/A	Y	N/A	Y
	18	Y	Y	Y	Y
	19	Y	N/A	Y	Y
	20	Y	N	Y	Y

其中，Y 表示检测到为共谋并给出警告，N/A 表示未检测到共谋未给出提示，N 表示未检测到共谋并给出错误提示。

5　结束语

针对 Android 共谋攻击隐蔽性好、检测效率差、检测准确率低的问题，提出基于组件通信的 Android 共谋攻击检测模型。模型分为训练阶段和检测阶段。训练阶段负责对训练样本的权限特征向量集学习和分类生成安全策略规则集，使用组件和通信方式特征向量集设计基于有限状态机的组件通信检测方法优化安全策略规则集；检测阶段负责将待测应用的特征向量集生成新权限状态机，通过匹配训练阶段的安全策略规则集完成共谋攻击的检测。实验结果表明所提模型具有较高的准确率和较好的检测效率。

未来考虑通过 Xposed 框架将本文所提模型整合到 Android 系统中，在真实环境下验证模型对共谋攻击的检测效果并对模型和方法进行持续改进。

参考文献：

[1] McaAfee Research Institute. McAfee labs threats report[R]. 2016: 1-53.

[2] FELT A P, WANG H J, MOSHCHUK A, et al. Permission re-delegation: attacks and defenses[C]//USENIX Security Symposium. 2011: 30-31.

[3] WU L, DU X, ZHANG H. An effective access control scheme for preventing permission leak in Android[C]//2015 International Computing, Networking and Communications Conference. 2015: 57-61.

[4] BLASCO J,CHEN T M. Automated generation of colluding apps for experimental research[J]. Journal of Computer Virology and Hacking Techniques, 2017, 36(17): 1-12.

[5] ARZT S, RASTHOFER S, FRITZ C, et al. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps[J]. ACM Sigplan Notices, 2014, 49(6): 259-269.

[6] ASAVOAE I M, NGUYEN H N, ROGGENBACH M, et al. Utilising semantics for collusion detection in Android applications[C]//International Workshop on Formal Methods for Industrial Critical Systems. 2016: 142-149.

[7] BOSU A, LIU F, YAO D, et al. Collusive data leak and more: Large-scale threat analysis of inter-app communications[C]//2017 ACM Conference on Computer and Communications Security. 2017: 71-85.

[8] WEI F, ROY S, OU X. Amandroid: a precise and general inter-component data flow analysis framework for security vetting of android apps[C]//2014 ACM SIGSAC Conference on Computer and Communications Security. 2014: 1329-1341.

[9] LI L, BARTEL A, BISSYANDE T F, et al. ApkCombiner: combining multiple android appsto support inter-app analysis[C]//IFIP International Information Security Conference. 2015: 513-527.

[10] SCHLEGEL R, ZHANG K, ZHOU X, et al. Soundcomber: a stealthy and context-aware sound trojan for smartphones[C]//The 2015 Network and Distributed System Security Conference. 2011: 17-33.

[11] BARTEL A, KLEIN J, LE TRAON Y, et al. Automatically securing permission-based software by reducing the attack surface: An application to android[C]//The 27th ACM International Conference on Automated Software Engineering. 2012: 274-277.

[12] SADEGHI A, BAGHERI H, MALEK S. Analysis of android inter-app security vulnerabilities using COVERT[C]//The 37th IEEE International Conference on Software Engineering. 2015: 725-728.

[13] MERCALDO F, VISAGGIO C A, CANFORA G, et al. Mobile malware detection in the real world[C]//ACM International Conference on Software Engineering. 2016: 744-746.

[14] KALUTARAGE H K, LEE C, SHAIKH S A, et al. Towards an early warning system for net work attacks using bayesian inference[C]//2015 IEEE 2nd International Conference on Cyber Security and Cloud Compu-

ting. 2015: 399-404.
[15] SCHAPIRE R E. Explaining adaboost[M]. Berlin, Germany: Springer-Verlag, 2013: 37-52.
[16] GILL A. Introduction to the theory of finite-state machines[J]. Mathematics of Computation, 1964, 92(29): 63-74.
[17] MCLLROY S, ALI N, HASSAN A E. Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store[J]. Empirical Software Engineering, 2016, 21(3): 1346-1370.
[18] CHO T, KIM H, LEE J, et al. A scheme for identifying malicious applications based on API characteristics[J]. Journal of the Korea Institute of Information Security and Cryptology, 2016, 26(1): 187-196.
[19] KIM H, CHO T, AHN G J, et al. Risk assessment of mobile applications based on machine learned malware dataset[J]. Multimedia Tools and Applications, 2017,35(23): 1-16.
[20] AGRAWAL A, SIMON G, KARSAI G. Semantic translation of Simulink/Stateflow models to hybrid automata using graph transformations[J]. Electronic Notes in Theoretical Computer Science, 2004, 109(11): 43-56.
[21] DHAVALE S, LOKHANDE B. Comnoid: information leakage detection using data flow analysis on Android devices[J]. International Journal of Computer Applications, 2016, 134(7): 1-18.
[22] OCTEAU D, LUCHAUP D, DERING M, et al. Composite constant propagation: application to android inter-component communication analysis[C]//The 37th International Conference on Software. 2015: 77-88.
[23] BOSU A, LIU F, YAO D D, et al. Collusive data leak and more: large-scale threat analysis of inter-app communications[C]//The 2017 ACM on Asia Conference on Computer and Communications Security. 2017: 71-85.
[24] GORDON M I, KIM D, PERKINS J H, et al. Information flow analysis of Android applications in DroidSafe[C]//2015 Network and Distributed System Security Conference. 2015: 1-16.

[作者简介]

杨宏宇（1969-），男，吉林长春人，博士，中国民航大学教授，主要研究方向为网络信息安全、移动系统安全等。

王在明（1990-），男，山东临沂人，中国民航大学硕士生，主要研究方向为移动系统安全。