doi:10.11959/j.issn.1000-436x.2017271

基于多层 SimHash 的 Android 恶意应用程序检测方法

陈波，潘永涛，陈铁明
（浙江工业大学计算机学院，浙江 杭州 310023）

摘 要：提出一个基于多层 SimHash 的相似度检测方法，通过对 APK 文件进行分析，最终从 5 个方面提取分析内容来表征 APK，同时在每一层上使用改进的 SimHash 方法进行相似度检测分析。通过从 APK 文件中提取的 AndroidManifest.xml 文件、从 dex 反编译得出的 Smali 代码累加和、Smali 文件指令提取、Java 代码集合、Java 指令集提取 5 个层面进行分析。同时通过学习 Voted Perceptron 投票算法，将其应用到检测过程中，采用信任值权重的方法，为每一层赋予一个可信值，并在最后得出结果时将每一层结果结合权重分析，实验分析结果表明该方法具有更好的检测效果。

关键词：Android；代码检测；SimHash；Voted Perceptron
中图分类号：TP393 **文献标识码**：A

Android malware detection method based on SimHash

CHEN Bo, PAN Yong-tao, CHEN Tie-ming
(College of Computer, Zhejiang University of Technology, Hangzhou 310023, China)

Abstract: A new similarity detection scheme based on hierarchical SimHash algorithm was proposed. The scheme extractd contents from different aspects to represent the APK file, then used the improved SimHash to respectively represent the file. The scheme analyzed the APK file by extracting the AndroidManifest.xml file in it, the sum of the Smali code from the decompilation of dex file, instructions extracted in Smali files, Java code set, and instructions extracted in Java code files. Through the study of Voted Perceptron voting algorithm, the scheme used trust weight method, by valuating a trust weight in every layer, then combined all the result with weight in every layer as a resule of scheme, the result can be more reasonable and more convincing.

Key words: Android, malware detection, SimHash, Voted Perceptron

1 引言

智能手机不断更新发展，不仅给人们带来了极大的方便与快乐，与此同时，也带来了一系列的安全隐患。隐私及敏感信息泄露、垃圾消息轰炸、恶意扣费、系统攻击等恶意行为不仅给使用者带来了使用上的不便，也给他们带来了经济上的损失。Android 智能手机占据了全球手机大半市场，因其开放性、开源性等特性被广大使用者所青睐，同时也因此被恶意应用程序所攻击。因此，开展对 Android 恶意应用程序的探索，寻求好的检测以及进行分类的方法具有一定的实际意义。

目前，对 Android 恶意应用程序，很多机构以及个人都从静态检测和动态检测进行了研究与分析。

在静态分析领域[1,2]，指的是在不运行应用程序的情况下，通过逆向工程技术，提取出应用程序中的特征然后基于对特征的分析检测应用程序。此领域的研究者一般通过 Android 应用程序的执行指令以及 APK 文件中反编译提取函数调用关系、权限等方面进行分析。

动态分析方法与静态分析方法不同，它是使用

收稿日期：2017-10-28
基金项目：国家自然科学基金资助项目（No.U1509214, No.6177202）；浙江省自然科学基金资助项目（No.LY16F020035）
Foundation Items: The National Natural Science Foundation of China (No.U1509214, No.6177202), The Natural Science Foundation of Zhejiang Province (No.LY16F020035)

虚拟机、沙盒等方法代替真机来安装 Android 应用程序，并在其之上模拟所安装的应用程序的实际运行状态，通过对应用程序在使用过程中的各种行为的监控来进行分析。这其中进行分析的数据，是通过手机应用程序在模拟运行时的行为记录提取出来的。动态分析主要是要求能模拟遍历用户操作，将应用程序的所有点都能够点击操作触发，力求更加真实地模拟用户的日常使用。另一方面是需要对 Android 运行环境进行有效的监控，包括流量监控、数据交互监控、敏感信息监控以及运行环境本身监控等，同时也需要对一些敏感的数据进行数据源跟踪等操作。

SimHash 相似度检测算法[3]源于 GoogleMoses Charikar 的文章——“Detecting near-duplicates for Web crawling”，其主要用途是检测和解决大规模网页的去重任务[4]。该算法使用先进的指纹数据来表征一整个文档，通过比较 2 个文档的指纹值之间的汉明距离来测量 2 个文档之间的相似度[5]。其基于指纹数据来进行相似度检测，在检测过程中能够减少计算成本，同时在一定程度上提高检测效率[6]。SimHash 的主要思想就是“降维”，通过将大量的文档信息转化为较短的一个指纹数据，最终只需要对指纹数据进行对比分析即可，通过指纹不仅能确定 2 个文档的内容是否相等，还能够比较 2 个文档的相似程度[7,8]。

Voted Perceptron 算法是一个新的、更简单的线性分类算法，该算法是在感知器算法的基础上，由 Helmbold 和 Warmuth 加入学习算法进行改进之后得到的。此外，Aizerman 等[9]所进行的工作表明，内核函数可以被有效地运用到算法中，并可在高维度的空间算法中有效地运行。该算法实现起来比较简单，其效果也比一般的支持向量机模型要好，因此，在高维度的空间中也非常适用。

2　SimHash 方法

相似度计算并不容易，因为对于电脑来说要识别相似但不相同的元素是比较复杂的。为了比较两样东西，人类大脑对每一项内容都创建了一系列标准，SimHash 所做的也是一样，它会使用每一个词作为描述文本内容的标准来对比 2 个文本，通过雅卡尔系数反映相似度。然而，这个系数并非十分有效，在第一个文本 A 中，需要检查 A 中每一个元素是否也在第二个文本 B 中。

SimHash 是建立在指纹过程之上的，它会建立指纹来对比文本。因此，就将会被其二进制指纹所替代，同时也更加有效[10]。

SimHash 算法的主要思想是首先定义一个 f 维度的空间，然后在这个空间中定义每一个特征所相对应的向量，之后将所有的向量结合各自的权重进行加权求和就得到了一个和向量作为结果。这样得到的一个和向量就表征了这个文档。而后，为了方便计算，因此将和向量进一步进行压缩转化，其规则为：对每一个向量得出一个相对应的 f 位签名信息，向量中维度的值大于 0 的，其在签名中所对应的位数就置 1，否则置 0。通过这样的方式进行转化，得到的签名信息就表征了此向量在各个维度中的值信息，同时一个 64 位的签名就可以表达多达 264 个维度，因此，根据向量在其各个维度信息也足以表征一个文档。

SimHash 算法的过程比较简单，SimHash 值的生成图解如图 1 所示。

从图 1 中可以简单明了地看出，SimHash 值的生成过程简单明了，大致过程如下。

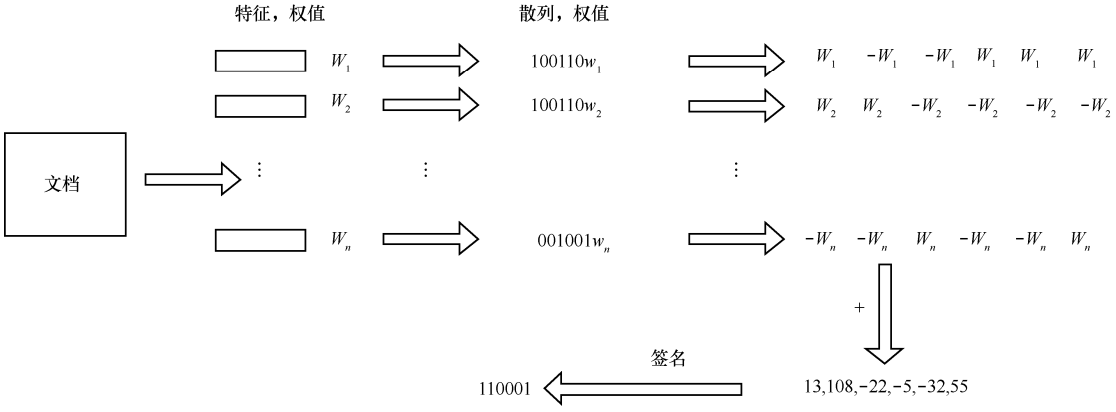

图 1　SimHash 值生成

步骤 1 确定 SimHash 值所需的位数，在这之中需要将存储的成本和需要处理的数据集大小一并考虑进去。

步骤 2 初始化一个 f 维的向量 $\boldsymbol{V}$，初始值为 $\boldsymbol{0}$；同时相对应地初始化一个二进制数 S，一共 f 位，值为 0。

步骤 3 提取原始文本的特征。

步骤 4 对于每一个特征，在分析时使用传统的散列方法对其计算得到一个 f 位的值 b，同时对 i=1 到 f

if (b 的第 i 位=1)

$\boldsymbol{V}$ 的第 i 个维度 = 该维度原来的值 + 该位的特征权重值；

else

$\boldsymbol{V}$ 的第 i 个维度 = 该维度原来的值 − 该位的特征权重值；

步骤 5 得到的最后结果 $\boldsymbol{V}$ 中，对其中的每一位进行转化处理

if（$\boldsymbol{V}$ 的第 i 个维度值>0）

S 的第 i 位=1；

else

S 的第 i 位=0；

步骤 6 最终得到的 S 作为签名输出。

该算法首先是在一个 f 维度的空间中为每个特征都建立一个相对应的向量，在这之中对其具体对应关系其实并不关注，只要在其建立的结果中不同的特征在此向量空间里所对应的向量是分散的，且相同的特征在空间中对应的是唯一的一个向量即可。

3 改进的 SimHash 方法

本文提出了一种新的改进的 SimHash 代码相似度检测方法。该方法在传统 SimHash 方法的基础上提出一种改进其准确性的方法，从而能够提高 SimHash 方法在检测相似度时的准确度。

整个改进的方法分为 6 个步骤，具体如图 2 所示。

步骤 1 分词

对于所提供的一个对象（文章、代码），通过将其分词，然后进行特征提取，这样就得到了可以使用的特征向量，接着给不同的特征向量赋予不同的权重值。

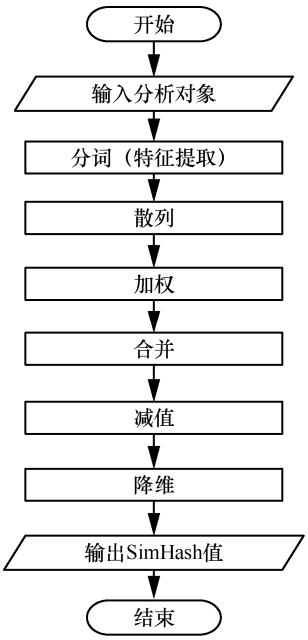

图 2 改进的 SimHash 算法步骤

步骤 2 散列

使用传统的散列函数，对每一个特征向量使用其得出相对应的散列值，这个值是由 0 和 1 组成的一个数串。

步骤 3 加权

在对每一个特征向量得到相应的散列值（*hash*）后，分别将其与在分词时赋予的权重值相结合，通过将其变换后与权重值相乘即可得到结果。其具体规则为

$$W = hash \cdot weight$$

其中，通过每一位是 1 还是 0 来判断其与权重值正相乘还是负相乘。

步骤 4 合并

将上一个阶段所得到的所有结果进行合并，合并成只有一个结果，这里与原算法一样只是简单累加。

步骤 5 减值

通过整理分析，选择一个阈值 T，将最后合并得到的结果序列串每一项减去设定的阈值 T，得到最终的结果序列串。

步骤 6 降维

与原 SimHash 方法相同，将通过减值操作所得到的序列串进行变换，对于序列串上面的每一位，若其为正数则将该位变为 1，否则将该位变为 0，这样就得到了所处理对象（文章、代码）的 SimHash 值。

其主要流程如图 3 所示。

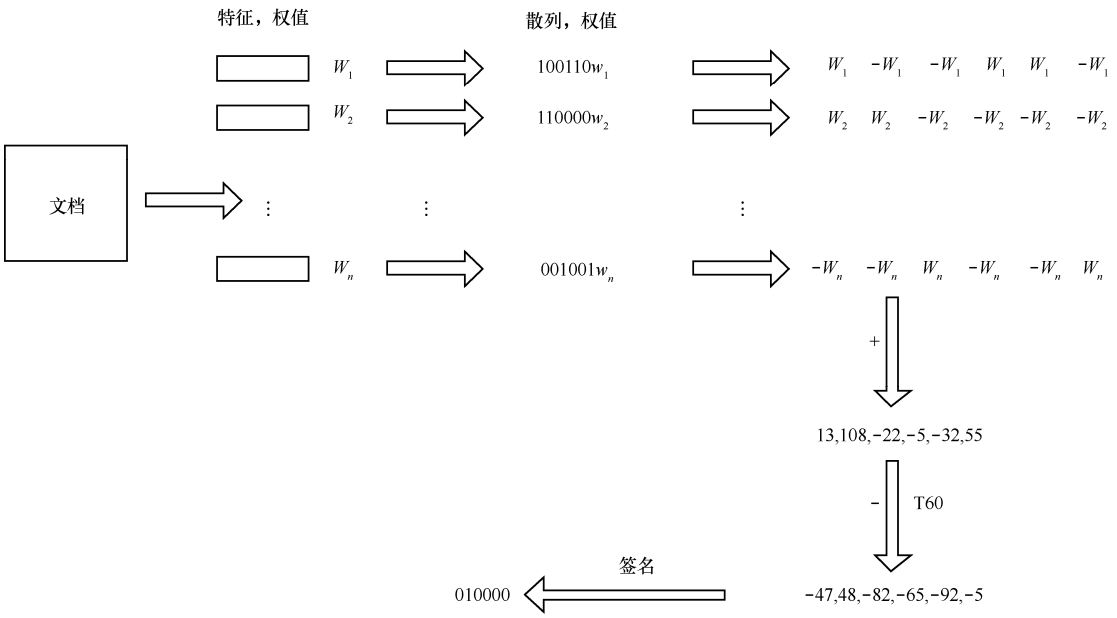

图 3　改进的 SimHash 值生成

该方法在传统 SimHash 方法的基础上，添加了一环，即在将所有特征向量进行转变、结合权重值累加等最后得到一个结果之后，再通过降维处理，最终得到所提供对象（文章、代码）的 SimHash 值。

以一句简单的英文语句为例，语句 1 为“Failure is probably the fortification in your pole”，语句 2 为“Failure a probably the fortification in your pocket?”，用 SimHash 原方法相比较两句语句时得到 2 个语句的相似程度为 1，即 100%。而用改进后的 SimHash 方法来测试 2 个语句的相似程度则不为 1。可见改进后的方案较之前的有较强的准确性。具体测试结果如图 4 所示。

```
old simhash
('s1:', 'Failure is probably the fortification in your pole.')
('s2:', 'Failure a probably the fortification in your pocket?')
14562932065
14562932065
('the similarity of two strings is:', 1.0)
new method
('s1:', 'Failure is probably the fortification in your pole.')
('s2:', 'Failure is probably the ticket in your pocket?')
14562899297
14831334496
('the similarity of two strings is:', 0.9819008060891353)
```

图 4　传统 SimHash 算法及改进后的算法效果对比

根据图 4 以及其他实验结果表明，经过改进后的 SimHash 方法相比于原来的 SimHash 方法在检测文章的相似度方面效果更加显著。在不同的地方也有体现，只要在使用时选好阈值 T，不断改进，就能在检测相似性时较 SimHash 方法更好的结果。

4　基于多层 SimHash 的 Android 恶意程序检测方法

Voted Perceptron 算法简介如下。

首先假设所有的实例都是点 x，其特征向量为 $\boldsymbol{X}$。人们使用 $|\boldsymbol{X}|$ 来表示 x 的欧几里得长度，x 的标签 y 的取值为 $\{-1, 1\}$。对大多数情况来讲，都假定标签 y 在范围内。

该算法的基础是 Rosenblatt 发明的感知器算法，这是在在线学习模型中研究的最自然的非常简单的算法。在线学习算法从一个初始化的预测向量 v=0 开始，它将每一个新的实例 x 的标签都预测为 $y' = \text{sign}(v.x)$。如果该预测的结果与标签 y 不相同，则将预测向量更新为 $v = v + x$。若该预测是正确的，则 v 不改变。接下去的实例都进行这样的步骤。

感知器算法应用于从一批训练实例集中基尼系那个学习的最常见的方式是通过不断地在训练集上重复运行算法，一直到它在所有训练集上都能够正确的预测向量。然后再将该预测规则应用于预测测试集的实例的标签之中。

本文就 APK 文件反编译、提取分析等从各个方面入手，从不同角度来表征 APK 文件，从而进行相似性分析，并在最后进行累加得到结果。本节提出的是 5 层比较可信的方面，即从 AndroidManifest.xml 文件、Smali 代码、Smali 代码指令提取、Java 代码以及 Java 代码指令提取 5 个方面。在研究中，还进

行了其他方面的研究测试，如 URL 提取、代码中变量名提取、分组名提取、指令数目比较等，最终选取如下 5 方面阐述。

在 Voted Perceptron 算法的基础上通过不断训练累积每一层的可信度，在进行相似度计算的时候，每一层得出的结果加上该层的可信度权值，来进行投票处理，根据投票结果来判断 Android 恶意代码的分类。这样不仅从多角度分析 Android 应用程序的代码，同时也对不同层分析的可信度做了加权处理，这样大大增强了结果的可信度。

修改后的算法如图 5 所示。

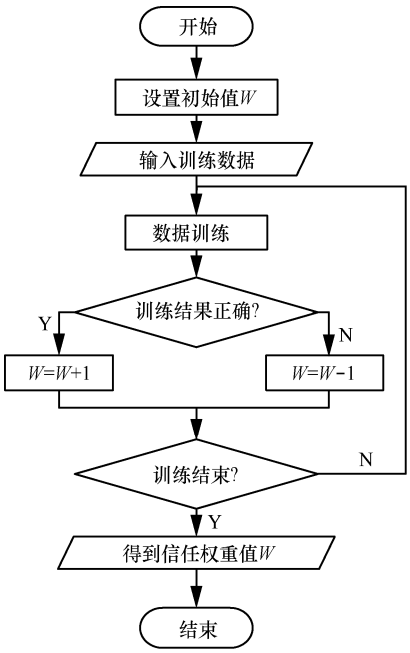

图 5 投票权重值训练流程

在每一层中，首先对该层赋予一个可信度初始值，在训练的时候测试每一种方法的实际准确性，在每一次的训练中，对每一层的训练结果做可信度权值增加。若该次训练该层检测结果正确，则该层的可信度权值加 1，反之，则将该层的可信度权值减 1。在经过足够多的训练结果之后，将对各层的可信程度有一定的结果。在测试时，则只需将每一层的测试结果与该层的可信度相乘，然后将 5 层的检测结构做投票统计，即可得到最终结果。

Android 恶意代码及其恶意家族分类训练过程如图 6 所示。

从图 6 中可以看到，通过上面提到的 5 层特征文件只用 SimHash 算法进行相似度分析，其将得到 5 个恶意家族分类模型。

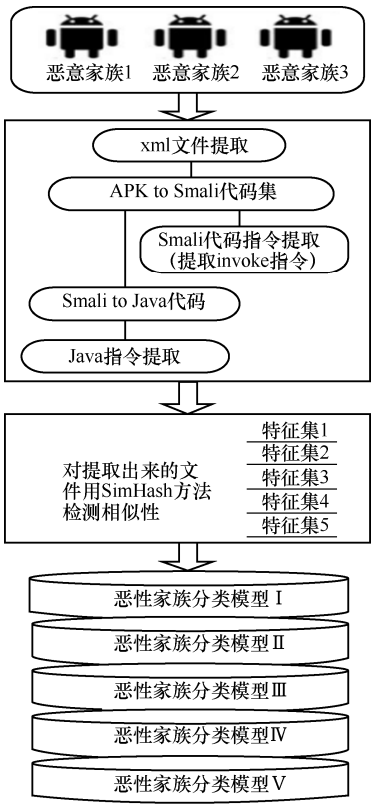

图 6 Android 恶意代码及其恶意家族分类训练过程

Android 恶意代码及恶意家族检测过程如图 7 所示。

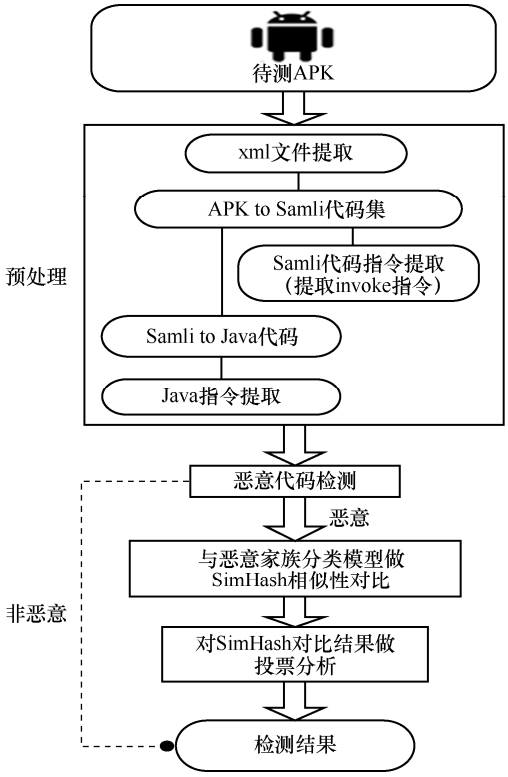

图 7 Android 恶意代码及其恶意家族分类检测过程

从图 7 中可以看到，通过对提交上来的 APK 文件进行预处理得到 5 层不同的内容，若检测出是恶意的应用程序，则可通过将其 5 层的内容通过与原来建立好的数据模型进行比较得出结论，该恶意应用程序属于哪一类。

算法举例如下。

假设每一层标识为 V_1、V_2、V_3、V_4、V_5，其各自分别的权重为 W_1、W_2、W_3、W_4、W_5，且权重初始值为 10。

L1：在训练阶段，对每一次的训练结果做累加处理，如第一次训练结果在各层的结果为：正确、错误、正确、正确、正确，则相应的每一层的权重作如下处理。

$$W_1 = W_1 + 1 = 11$$
$$W_2 = W_2 - 1 = 9$$
$$W_3 = W_3 + 1 = 11$$
$$W_4 = W_4 + 1 = 11$$
$$W_5 = W_5 + 1 = 11$$

在第二次训练的结果为：正确、错误、正确、错误、正确，则各权重处理如下。

$$W_1 = W_1 + 1 = 12$$
$$W_2 = W_2 - 1 = 8$$
$$W_3 = W_3 + 1 = 12$$
$$W_4 = W_4 - 1 = 10$$
$$W_5 = W_5 + 1 = 12$$

假设这样进行了 1 001 次训练过程（考虑到最后权重累加可能出现的结果不一致时出现的权重值对等情况，因此选用奇数次训练），得到最终各层的训练结果为：$W_1 = 525$，$W_2 = 313$，$W_3 = 887$，$W_4 = 355$，$W_5 = 749$。

L2：测试阶段，对待测试的 Android 恶意程序进行各层分层的测试，并将各层的结果乘以权重之后相加，得出最后的结论。注意，这里的相加并不是数值上的相加，其规则为：若测试结果一致的，该结果的权重数值相加，否则不做变化。如某个 Android 恶意程序各层测试的结果为：$S_1 = 1$，$S_2 = 3$，$S_3 = 1$，$S_4 = 2$，$S_5 = 1$，则先对该应用程序的一致的结果作权重累加处理，即结果为 1 的有 S_1、S_3 和 S_5，将其权重值相加，得到结果为 1 的权重为 2 161，另外没有相同的结果，不作其他处理。因此，对结果处理后得到该应用程序结论为：结果为 1 的可能性为 2 161，结果为 2 的可能性为 355，结果为 3 的可能性为 313。投票取大值，得出结论：该应用程序结果为 1。

5　Android 恶意代码检测与分类实验

如图 8 所示，Android 检测方法系统总体可分为样本分析、训练投票和应用程序检测 3 个模块。

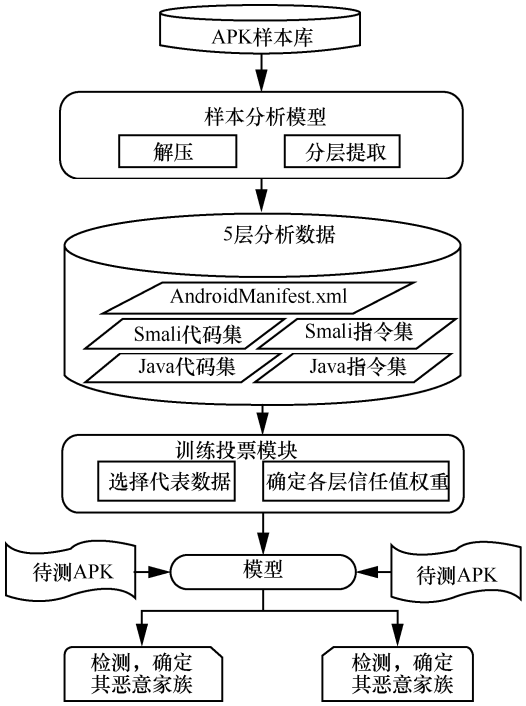

图 8　整体系统框架

1) 样本分析模块

从样本库得到恶意的 APK 文件的样本后，首先进行解压缩操作，以及利用 apktool 工具对其进行反编译，从中得到所需要的 AndroidManifest.xml 文件（第一层）以及 Smali 代码文件夹。然后利用自己编写的脚本程序对 Smali 代码文件夹中所有 Smali 文件提取内容整合到一个文件中（第二层），之后对此文件作指令提取，将所有的方法指令（method）提取出来，在整合到另一个文件中（第三层），同时利用脚本程序调用 dex2jar 工具对 dex 文件进行反编译得到 jar 包，然后再利用 jad 工具将 jar 包里面提取出来的所有 class 文件反编译成为 Java 文件，之后将所有 Java 文件的内容整合到一和文件之中（第四层），最后在所得到的所有 Java 代码的文件中进行指令提取，将所得到的所有指令都整合到一个文件中（第五层）。

2) 训练投票模块

该模块从前一个模块中得到的 5 层内容进行训练分析，首先是同一个恶意家族里面的 APK 文件所提取出来的内容进行分析，不断累计更新每一层

的信任权重，得到初步合适的权重值。在检测的同时选取恶意家族中的有代表的 APK 文件，即检测结果都相近的 APK 文件。其次再选取不同恶意家族中的样本进行检测对比，从反面迭代更新每一层的信任权重，与此同时对前面选出的不同恶意家族的代表 APK 文件进行筛选，提出不同恶意家族的但是检测出相似度较大的 APK 文件。最终得到一个 5 层的恶意样本库，以及投票算法所需的每一层的信任权重值。

3) 应用程序检测模块

使用样本库外的 APK 文件作为输入，进行恶意家族分析，或挑选 2 个 APK 文件作为输入，测试 2 个 APK 文件的相似程度。最终将得到一个结果，或该恶意 APK 文件比较可能属于哪一类恶意家族，或是输入的 2 个 APK 文件的相似度结果作为输出。

6 结束语

针对 Android 恶意应用程序的检测、SimHash 相似度检测算法精确度不足等问题以及 Voted Perceptron 算法的研究，一方面，对原 SimHash 方法进行了改进，更加能得到 2 个检测文本之间的准确相似度。另一方面，提出了一个基于层次式的相似度检测方法，通过不同层面的检测得出的结果综合考虑分析。通过对 Voted Perceptron 算法进行学习，将其变换应用到本文中，提出基于权限的投票算法，将每一层的结果带上权重输出，使结果具有更好的效果。

本文提出的基于层次式的检测方法能够很好地将恶意应用程序进行分类，但是仍然存在一些需要修改和关注的问题：1) Android 应用程序里面还可以提取很多信息，有些恶意应用程序只是通过 URL 联网时来进行攻击，有些是在动态运行时才触发恶意行为，如果能将静态分析与动态行为分析相结合起来进行综合分析，这样将更全面，其得到的结果也会更加精确。2) 本文在实际选择层次式分析的时候所选取的是有限的 5 个层次，其他一些对比层次应用其他的对比方法可能具有相同或更好的结果，还有待挖掘开发。3) 本文所实验的次数还是比较少，因此，对于改进的 SimHash 方法中阈值的选取以及分层检测模型中各层的信任值的界定略显粗略，不断地进行实验，对恶意应用程序的检测分析，不断更新参数值，将对检测的结果有一定的提高。4) 本文所选取的训练测似乎的实验数据为几年前的恶意样本库，若不断扩大训练测试数据集，不断更新实验数据，不断更新检测模型，将得到更好的检测结果。

因此，为了进一步提高检测效率，可以考虑将数据处理和分类改为并行化处理，并使用更大的一个数据集合进行训练和测试，同时使用其他特征信息展开研究分析。

参考文献：

[1] YAN Q, LI Y, LI T, et al. Insights into malware detection and prevention on mobile phones[C]//Security Technology - International Conference, Sectech 2009, Held As. DBLP, 2009:242-249.

[2] CHANDRAMOHAN M, TAN H B K. Detection of mobile malware in the wild[J]. Computer, 2012, 45(9):65-71.

[3] SOOD S. Probabilistic SimHash Matching[J]. 2011.

[4] UDDIN M S, ROY C K, SCHNEIDER K A, et al. On the effectiveness of SimHash for detecting near-miss clones in large scale software systems[C]//Working Conference on Reverse Engineering. IEEE, 2011: 13-22.

[5] 余意, 张玉柱, 胡自健. 基于 SimHash 算法的大规模文档去重技术研究[J]. 信息通信, 2015(2):28-29.

[6] BUYRUKBILEN S, BAKIRAS S. Secure similar document detection with SimHash[M]//Secure Data Management. 2014:61-75.

[7] MANKU G S, JAIN A, SARMA A D. Detecting near-duplicates for web crawling[C]//International Conference on World Wide Web. ACM, 2007:141-150.

[8] 周龙泉, 卫文学. 基于主成分分析与 SimHash 的入侵检测方法[J]. 计算机与数字工程, 2015(7):1291-1294.
ZHOU L Q, WEI W X. Intrusion detectin method based on principal component analysis and SimHash[J]. Computer and Digital Engineering, 2015(7):1291-1294.

[9] FREUND Y, SCHAPIRE R E. large margin classification using the perceptron algorithm[J]. Machine Learning, 1999, 37(3):277-296.

[10] UDDIN M S, ROY C K, SCHNEIDER K A, et al. On the effectiveness of SimHash for detecting near-miss clones in large scale software systems[C]//Working Conference on Reverse Engineering. IEEE, 2011: 13-22.

作者简介：

陈波（1971-），男，浙江慈溪人，浙江工业大学副教授，主要研究方向为无线和移动安全。

潘永涛（1991-），男，浙江绍兴人，浙江工业大学硕士生，主要研究方向为网络安全、大数据。

陈铁明（1978-），男，浙江诸暨人，浙江工业大学教授、博士生导师，主要研究方向为网络空间安全。