基于权限和API特征结合的Android恶意软件检测方法

邵舒迪[1,2] 虞慧群[1] 范贵生[1]
(华东理工大学信息科学与工程学院 上海 200237)[1] (上海市计算机软件评测重点实验 上海 201112)[2]

摘 要 随着Android操作系统的广泛应用,基于Android平台的应用程序的数量日益增长。如何有效地识别恶意软件,对保护手机的安全性至关重要。提出了基于权限和API特征结合的Android恶意软件检测方法,该方法通过反编译apk文件来提取权限特征和API特征,并将两者相结合作为一个整体的特征集合。在此基础上,采用分类算法进行恶意软件的甄别。实验结果表明,该方法的判别准确率高于权限集合或API集合单独作为特征的判别方法,从而能更加有效地检测Android恶意应用程序。

关键词 API,权限,特征集合,Android应用,恶意软件检测
中图法分类号 TP309 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.029

Detecting Malware by Combining API and Permission Features

SHAO Shu-di[1,2] YU Hui-qun[1] FAN Gui-sheng[1]
(School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)[1]
(Shanghai Key Laboratory of Computer Software Testing and Evaluating, Shanghai 201112, China)[2]

Abstract With the use of Android OS, the number of Android applications is getting larger and larger. Therefore, how to detect malware is very important for protecting the mobile phone security. In this paper, we extracted API feature and permission feature by reverse-engineering the apk files respectively. Then, the two features are combined into a feature set. Finally, with different classification algorithms, the malwares can be detected. As a result, compared to single API or permission feature, higher detecting accuracy is gotten, which shows that the feature combination of permission and API is more efficient in detecting malicious Android applications.

Keywords API, Permission, Feature set, Android applications, Malware detection

1 引言

随着手机硬件性能的不断发展,智能手机在人们日常生活中变得越来越普及。目前市面上的智能手机操作系统主要有Android,iOS,Windows Phone和Blackberry OS。根据IDC[1]的相关统计,在2015年的第一个季度,市面上Android操作系统的占有份额高达78.0%,远远高于iOS和Windows Phone等其它几个操作系统。Android操作系统的不断发展壮大,自然而然促使了Android应用程序的日益扩张。根据AppBrain网站[2]的调查,在2014年Google Play市场中的应用程序数目增长了近20万个,平均每天有500多个新程序进入应用市场。除去Google Play市场,市面上还有许多其他的第三方应用程序下载平台。但是相关部门的监管力度往往有限,导致一些恶意应用程序流入市场,通过人们的下载疯狂传播,给人们造成了巨大的损失。因此,如何能够有效地从繁多的程序中检测出恶意应用程序,以保护Android手机用户的安全与利益,对研究人员来说是一个必要而又迫切的挑战。

本文针对Android应用程序的权限机制和高危API的调用,提出权限和API特征结合的方法。首先,提取应用程序中的权限集合和高危API集合作为特征;然后,将权限特征和API特征相结合,形成新的API-权限特征集合;最后,结合分类算法,进行恶意应用程序与普通应用程序的甄别。实验结果表明,与单独采用权限或API作为特征的判别方法相比,本文方法能更加精确地区分出恶意软件,更全面地体现出应用程序的特征。

本文第2节介绍相关工作;第3节介绍权限和API特征相结合的方法;第4节通过仿真实验说明方法的有效性;最后总结全文。

2 相关工作

目前,关于Android恶意应用程序的检测主方法较多,但总体而言,其大致是围绕权限机制和API的调用这两大方面来进行展开的。

权限机制是由Google提出的一种安全机制,用于应用程

到稿日期:2015-11-30 返修日期:2016-02-24 本文受国家自然科学基金(60905043,61073107,61173048),高等学校博士学科点专项科研基金(20130074110015),中央高校基本科研业务费专项基金(WH1314038)资助。
邵舒迪(1991—),男,硕士生,主要研究方向为软件工程、恶意软件检测,E-mail:ssddsg@163.com;虞慧群(1967—),男,博士,教授,博士生导师,CCF高级会员,主要研究方向为软件工程、可信计算、形式化方法;范贵生(1980—),男,博士,副教授,主要研究方向为软件工程、面向服务计算、形式化方法分析与验证。

序间的组件访问以及对一些安全敏感项的限制。文献[3]的研究表明，基本上所有用户并不了解权限机制存在的意义，所以在面对恶意软件时，应用程序的用户们往往并不能做出正确的安全决策，从而给自己移动设备上的数据及隐私带来了很大的安全隐患。文献[4]发现大多数应用程序中往往只会有一小部分的权限会频繁出现，而 Google 为 Android 操作系统量身定制的一整套权限中绝大部分并不经常出现。这些经常出现的权限可以作为人们研究的对象，以区分出异常的应用程序。文献[5]提出了一款名为 Stowaway 的工具，在这款工具的帮助下，人们可以在 Apps 的开发过程中识别程序员是否存在过度申请权限的行为，因为这种看似不起眼的行为将会为应用程序埋下诸多安全隐患。而文献[6]则是将每一条权限看作一个特征，以此建立特征向量，通过分类算法区分恶意程序和正常程序。但是只将权限作为特征存在局限性，因为这无法完整地描述出恶意软件的特性。

本文所指的 API 为 Android 系统本身所提供的 API，属于 Android 体系结构中的应用程序框架层。应用程序可以通过对 API 的调用，来实现诸如获取和泄露隐私信息、发送扣费短信或非法接入因特网等恶意行为。文献[7]为了检测 phoneID 的信息泄露，在特定的 API 中插入了 log. v 方法来进行监控和观测，但是只关注了部分的 API，具有一定的局限性。文献[8]通过提取应用程序中所调用的 API 及其参数等信息作为依据，来判断该软件是否为恶意程序。文献[9]根据文献[5]提供的 API-Permission 映射关系，通过动态分析的方法记录下应用程序在实际运行中所涉及的所有权限，对比静态分析方法预先获得的权限集合来判断应用程序是否有过度申请权限的现象。

3 权限和 API 特征结合的恶意软件检测方法

权限和 API 特征的结合就是将权限的特征集合与 API 的特征集合相结合，而非单独使用任意一项集合作为特征来分类识别恶意应用程序和非恶意应用程序。

为了获得应用程序的 API 和权限的特征集合，本文使用 apktool 这款 Google 开源工具来反编译 apk 文件，获得 Manifest 文件、资源文件和签名文件等。获取权限和 API 的过程以及 apk 文件的反编译结果如图 1、图 2 所示。

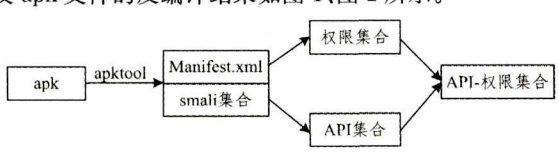

图 1 获取 API 和权限的过程

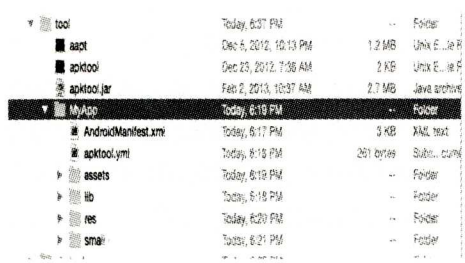

图 2 apk 文件的反编译结果

在 Manifest. xml 文件和 smali 文件中提取需要的权限和 API，生成相应的特征集合；再将两项特征结合，即可生成 API-权限特征集合。

3.1 特征选取

apk(application package file)，即 Android 应用程序包文件，是一种 Android 操作系统上的应用程序安装文件格式。一个 apk 文件所包含的内容有：被编译的代码文件(即. dex 文件)、资源文件(即 resources)、assets、证书文件(即 certificates)和清单文件(即 manifest. xml)等。从结构上来说，apk 是一种基于 zip 文件格式的文件，它与 jar 文件的构造方式相似[10]。

本文用到的权限信息和 API 信息分别存储于 Manifest. xml 文件和 smali 文件。为了取得上述信息，通过反向工程的方式，使用由 Google 公司提供的免费软件 apktool 解压缩 apk 文件，获得实验需要的 manifest. xml 文件和 smali 文件，从中提取出权限特征集合和 API 特征集合。

本文所涉及的基于特征的检测方法的实质就是对应用程序进行二分类，通过对程序的特征进行分析，把它归类为正常应用程序或者恶意应用程序。分类过程中，将作为分类依据的要素称为特征，而特征的选取决定了分类的准确性和效率。

(1)权限。Android 设立权限机制的主要目的是对应用程序可以执行的某些具体操作进行权限细分和访问控制。开发人员在开发应用程序时，只有在 Manifest. xml 文件中申请了该应用程序所需要的全部权限信息，这款应用软件才能被正常地安装与使用。Google 为 Android 系统本身共设立了 134 个权限，同时允许用户进行自定义权限的操作。本文只研究 Android 本身所提供的那部分权限。文献[15]研究发现，在普通应用程序和恶意应用程序的权限使用频率统计中，与手机 SMS 信息相关的一些权限(62. 7%)以及开机自启动(54. 6%)等权限在恶意应用程序中往往会被广泛使用。这些权限信息上的差异为权限可以成为 Android 恶意应用程序检测的一项特征提供了理论上的可行性。

(2)API。本文所研究的 API 是指由 Android 系统本身所提供的函数接口。通过这些函数的调用，应用程序即可访问和获取手机中的一些敏感数据，诸如联系人、地理位置、照片和账号等；或触发一些高危行为，如偷偷接入网络和发送扣费短信等。本文把这些涉及敏感数据和高危行为的 API 称作为高危 AP。与权限信息一样，这些 API 在使用情况上会因为正常软件与恶意软件的不同而存在巨大差异。在一些高危 API 的调用次数上，恶意应用远远超过普通应用，这能够在一定程度上反映出一款应用程序真实的行为特点，因此可以作为识别应用程序恶意行为的一个特征依据。

3.2 API-权限特征集合

如上文所述，权限信息和 API 信息都能一定程度地反映一款程序的特点，因此都可以单独作为特征向量使用，再结合分类算法，进而可以甄别恶意软件和正常软件。但其存在一定的局限性，对分类结果的准确性会造成一定的影响。比如，在实际开发过程中，程序员们往往会过度地申请权限，导致很多权限在应用程序中并未被使用，或代码中所包含的这些 API 在程序的实际运行过程中可能也并未真正得到调用。这

些行为都会对实验的准确性造成一定的干扰。因此，本文把权限特征和 API 特征相结合，生成一个结合的特征，以更好地表现软件的特征。获得 API-权限特征集合的步骤如下。

(1)提取权限特征。本文采用文献[6]中所提到的互信息的方法来衡量具体的权限与应用程序之间的相关性，并根据相关性的高低来选择权限集合。互信息的计算如式(1)所示：

$$I(X,C)=\sum_{X_i}\sum_{C_j}p(X=x_i,C=c_j)\times\log\frac{p(X=x_i,C=c_j)}{P(X=x_i)p(C=c_j)} \tag{1}$$

其中，变量 X 表示权限在一款应用中是否出现的情况(出现或未出现)，变量 C 代表应用的类别(属于正常软件或者恶意软件)，$P(X=x_i)$表示变量 X 值为 x_i 的概率，$P(C=c_i)$表示变量 C 的值为 c_i 的概率，$P(X=x_i,C=c_j)$表示变量 X 的值为 x_i 而变量 C 的值为 c_i 的频率。根据互信息公式，求出每个权限与软件的相关性数值，该值的取值范围为 0～1，值越大，说明两者的相关性越高，值为 0 表示两者间不存在相关性，值为 1 表示两者必然存在相关性。取出 30 个相关性最高的权限作为特征，每个应用程序可用一个 30 维的向量 $[Per]_{1\times30}$来表示，每个维度对应一个权限。如果在程序的 Manifest. xml 文件中含有该权限，则用 1 表示，若没有则用 0 表示。相关性最高的 10 项权限如表 1 所列。

表 1　相关性最高的 10 项权限

编号	权限
1	SEND_SMS
2	RECEIVE_SMS
3	READ_SMS
4	READ_PHONE_STATE
5	READ_EXTERNAL_STORAGE
6	WRITE_SMS
7	SET_ALARM
8	RECEIVE_BOOT_COMPLETED
9	RECEIVE_WAP_PUSH
10	WAKE_LOCK

(2)提取 API 特征。根据 Android 官网所提供的 API 信息[11]和常见的窃取敏感信息、网络流量和恶意扣费等恶意软件行为，选取其中 20 个常见的高危 API 作为特征集合，每个应用程序可用一个 20 维的向量 $[API]_{1\times20}$来表示，每个维度对应一个 API。遍历程序的所有. smali 文件，统计每个 API 被调用的次数作为向量的值。部分较常见的 API 如表 2 所列。

表 2　部分常见的高危 API

编号	权限
1	getSubscriberId( )
2	getDeviceId( )
3	getNETWORKCountryIso( )
4	getLatitude( )
5	getLogitude( )
6	getOutputStream( )
7	getInputStream( )
8	sendTextMessage( )
9	sendDataMessage( )
10	startService( )

(3)将权限特征与 API 特征相结合。根据步骤(1)和步骤(2)，每款应用软件都能获得一组权限特征向量 $[Per]_{1\times30}$和一组 API 特征向量 $[API]_{1\times20}$。将两组向量结合，每个应用程序可用一个 50 维的向量 $[API,Per]_{1\times50}$来表示。结合后的新特征能更好地表现应用程序的特点。

3.3　方法评价

用 Acc，TPR，FPR 和 G 即准确率、真正类率、负正类率和 G-Mean 值作为方法的评价指标。Acc 表示所有被正确分类应用和总应用数的比例；TPR 表示所有被正确分类的恶意应用程序数和被分类为恶意应用程序总数的比例；FPR 表示被错分成正常程序的恶意应用程序总数和被分类为正常应用程序总数的比例。Acc 与 TPR 的值越高，FPR 值越低，则分类效果越好。在实际的分类中，Acc 只能反映分类器对数据集的整体分类性能，但不能正确反映不平衡数据集的分类性能，针对不平衡数据，需提出更为合理的评价标准。G-mean 是不平衡数据集学习中常用的评价准则，G 值越大越能合理地评价不均衡数据集的总体分类性能。

$$Acc=\frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

$$TPR=\frac{TP}{TP+FN} \tag{3}$$

$$FPR=\frac{FP}{FP+TN} \tag{4}$$

$$G=\sqrt{TPR\times(1-FPR)} \tag{5}$$

式(2)－式(5)中，TP 表示恶意应用软件被正确分类的个数，TN 表示正常应用软件被正确分类的个数，FN 表示恶意应用软件被错误分类的个数，FP 表示正常应用软件被错误分类的个数。

4　实验结果

4.1　实验准备

本次实验选取了 100 个恶意软件和 500 个正常软件作为实验数据集，其中 100 个恶意软件来自于 VirusShare. com[12]所提供的恶意软件样本集；而 500 个正常应用程序则来源于第三方应用程序，通过网络爬虫程序收集获得。所有的程序都经过提供恶意软件分析服务的 VirusTotal[13]网站检测验证，从而确保恶意软件和正常软件不会混淆，不会影响到实验结果。本次实验在 CPU 为 Intel(R) Core(TM) i5、内存为 8GB 的笔记本计算机上运行，实验工具为 Matlab_R2012a。

4.2　分类算法

分类算法的作用是根据所得到的特征向量来区分样本的不同类别。本文用到的分类算法有 SVM 和 K-近邻(KNN)算法。

(1)支持向量机(Support Vector Machine，SVM)是由 Vapnik 等人提出的一种分类方法，它的主要依据是统计学中的 VC 维理论和结构风险最小原理。它的本质是通过最大化分类间隔构造出最优分类超平面来获得良好的泛化能力，能较好地解决非线性、高维数和局部极小点等一系列问题。简单来说，对于分类问题，支持向量机就是利用一个非线性的映射将原始数据映射到高维层次；然后在新的维度上计算该区域上最优的决策曲面，由此确定该区域中未知样本的类别。

(2)K-近邻(K-Nearest Neighbors，KNN)算法的思路是：对于特征空间中的一个样本，离它最近的 k 样本大多属于某

一个类别，那么该样本也属于这个类别。该算法的执行步骤如下：

输入：n 个训练样本在空间中的坐标 A[n]，近邻数 k，测试样本 x

输出：x 所属的类别

步骤 1 计算出测试样本 x 与训练样本 A[i]间的欧氏距离 d(x，A[i])，i=1，2，…，n；

步骤 2 将 d(x，A[i])按升序排列，取前 k 个距离最小的样本记为 dis[1]…dis[k]；

步骤 3 统计 dis[1]…dis[k]中每个样本所属的类别；

步骤 4 具有最多样本数的类别即为样本 x 的类。

4.3 实验结果

首先分别提取每款应用中的权限特征和 API 特征，形成两个特征向量；然后将这两种权限特征合并，得到每款应用程序的权限和 API 的混合特征向量；最后利用主成分分析方法 PCA 将权限和 API 的混合特征向量进行降维处理，去除冗余信息，得到降维后的 API 和权限的混合特征向量。对于这 4 组向量，本文分别使用基于径向基函数(RBF)的 SVM 算法和 KNN 算法进行分类，测试集和训练集的分配比例为 1 ∶3，每个算法运行 5 轮之后取平均值。使用 SVM 算法得到的实验结果如图 3 所示。

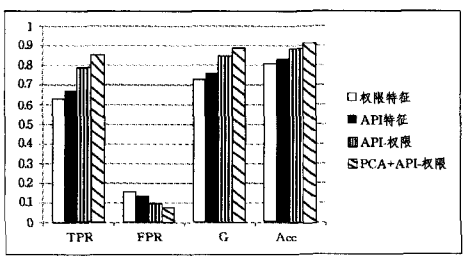

图 3 SVM 算法的实验结果

从图 3 可以看出，对于 SVM 算法，无论将权限信息还是 API 信息作为单一的特征来对 Android 恶意应用程序进行检测，其分类的效果都差于将权限和 API 的特征相结合的检测方法。在将两项特征结合之后，实验结果的各项指标均有提升。其中，Acc 和 G-mean 的值均在 90%左右，表明本文的方法取得了不错的实验结果。

同样地，使用 KNN 算法得到的实验结果如图 4 所示。

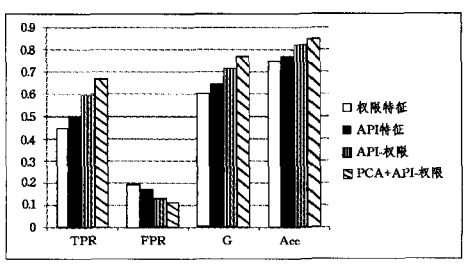

图 4 KNN 算法的实验结果

对于 KNN 算法来说，由于其算法本身比较简单，因此在面对不均衡的数据集时，会对实验结果产生一定的影响。由图 4 可以看出，KNN 算法产生的各项数据均差于 SVM，并且 G-mean 值都未超过 80%，但从其整体上来说，API-权限特征结合的检测方法与另外两种使用单独特征的检测方法相比，各项指标均有提高，这客观地说明，对于本次实验中所用的不均衡数据集，本文方法能取得较好的分类效果。

总体来说，由于恶意应用程序中的恶意行为多样化，与单独使用一项特征的检测方法相比，API-权限特征结合的检测方法能够更好且更有效地表现出 Android 应用程序的特征。在对 API-权限结合的特征向量使用主成分分析方法 PCA 进行降维之后，部分冗余特征被去除，分类的准确率变得更高，取得了更好地检测效果。

结束语 本文从 Android 应用程序的 API 和权限这两项特征出发，通过反向编译收集应用软件样本集，从中提取出高危 API 特征和权限特征，将两者结合成 API-权限特征集合，通过不同的分类算法来实现 Andriod 恶意应用程序的检测。对 100 款恶意程序和 500 款正常程序进行了仿真实验，结果验证了本文所提出的将 API 和权限相结合的判别方法的高效性。实验结果表明，本文所提方法能够有效地提高 Andriod 恶意应用程序检测的准确率，更全面地体现 Android 应用的特征。

参 考 文 献

[1] Smartphone OS Market Share[OL]. http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[2] Googel Play States[OL]. http://www.appbrain.com/stats/stats-index.

[3] FELT A P, HA E, EGELMAN S, et al. Android permissions: User attention, comprehension, and behavior[C]// Proceedings of the 8th Symposium on Usable Privacy and Security. 2012: 1-14.

[4] BARRERA D, KAYACIK H G, VAN OORSCHOT PC, et al. A methodology for empirical analysis of permission-based security models and its application to android[C]// Proceedings of the 17th ACM Conference on Computer and Communications Security. 2010: 73-84.

[5] FELT A P, CHIN E, HANNA S, et al. Android permissions demystified[C]// Proceedings of 18th ACM Conference on Computer and Communications Security. 2011: 627-638.

[6] WANG W, WANG X, FENG D W, et al. Exploring Permission-Induced Risk in Android Applications for Malicious Application Detection[J]. IEEE Transaction On Information Forensics and Security, 2014, 9(11): 1869-1882.

[7] NISHIMOTO Y, KAJIWARA N, MATSUMOTO S, et al. Detection of Android API Call Using Logging Mechanism within Android Framework[C]// International Conference on Security and Privacy in Communication Systems. 2013: 393-404.

[8] AAFER Y, DU W L, YIN H. DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android[C]// Proceedings of International Conference on Security and Privacy in Communication Networks. 2013: 86-103.

[9] GENEIATAKIS D, FOVINO I N, KOUNELIS I, et al. A Permission verification approach for android mobile applications [J]. Computers & Security, 2015, 49: 192-205.

[10] APK[OL]. https://zh.wikipedia.org/wiki/APK.

[11] Package Index[OL]. http://developer.android.com/reference/packages.html.

[12] Virusshare[OL.]http://virusshare.com.

[13] Virustotal[OL]. https://www.virustotal.com.

[14] ZHANG R. Research on Malware Detecting based on Static Analysis under Android Environment [D]. Chongqing: Chongqing University, 2014. (in Chinese)
张锐. Android 环境下恶意软件静态检测方法研究[J]. 重庆：重庆大学，2014.

[15] ZHOU Y J, JIANG X X. Dissecting android malware: Characterization and evolution[C]// Proceedings of the IEEE Symposium on Security and Privacy. 2012: 1063-1069.

[16] XING L Y, PAN X R, WANG R, et al. Upgrading your android, elevating my malware: privilege escalation through mobile os updating[C]// Proceedings of the 35th IEEE Symposium on Security and Privacy. 2014: 393-408.

[17] PANDITA R, XIAO X S, YANG W, et al. Whyper: towards automating risk assessment of mobile applications[C]// Proceedings of the 22nd USENIX Conference on Security. 2013: 527-542.

[18] WERTHMANN T, HUND R, DAVI L, et al. Psios: bring your own privacy and security to ios devices[C]// Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. 2013: 13-24.

(上接第 134 页)

的演化方法将原来的编解码器演化成新编解码器的整个过程用时 23s，主要用时在于手动输入内容。如果重新开发编解码器，一个技能娴熟并且对测试数据非常了解的测试人员也需要至少 5min 的时间，而且开发出来的编解码器一般都会存在一些问题，需要经过多次调试后才能正常使用。编写和调试的耗时一般在 10～30min。当测试数据比较复杂时，还会花费更多的时间。假设一个软件系统中有 20 个功能因为演化导致原来的编解码器不可用，若使用本文方法，只需要 10min 的时间即可得到所有的可用编解码器，但是手动开发至少需要 200～300min，本文方法将速率提高了 20～30 倍。被测系统的规模越大，尤其是中途更换测试人员时，这种从原编解码器进行演化以替代重新开发的速度优势也越明显。

结束语　软件演化表明了软件适应各种变化的能力。本文提出了一种对 TTCN-3 编解码器进行演化的方法，并且对其中的演化算法进行了实现，最后对实现结果进行了验证。该方法将不能继续使用的编解码器自动演化成了可用的编解码器，为 TTCN 测试工具软件的使用提供了便利，节省了测试成本。但是本文的演化方法仍然存在不足之处，即演化后的编解码器程序不存在循环结构，对于结构相同且可以用循环语句表示的编解码程序，这里也需逐条写出，所以当测试套中定义的数据类型比较庞大时生成的编解码器会相对臃肿。调整编解码器的程序结构是下一步的研究目标。

参 考 文 献

[1] LEHMAN M M, PERYY D E, RAMIL J F. Metrics and Laws of Software Evolution-the Nineties Views[C]// Proceeding of 4th International Symposium on Software Metries, 2000. IEEE Computer Society Press, 2000: 20-32.

[2] MENS T, BUCKLEY J, ZENGER M, et al. Towards a Taxonomy of Software Evolution[C]// International Workshop on Unanticipated Software Evolution, 2003. Warsaw, Poland, 2003: 34-45.

[3] ETSI ES 201 873-1 v4.5.1. Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language[S]. 2013.

[4] SUN M, ZOU F Y, MA J, et al. A method to automatically generate codec based TTCN-3 to software testing[C]// IEEE 2nd International Conference on Communication Software and Networks, 2010. Singapore: IEEE Press, 2010: 200-204.

[5] RAJLICH V, SILVA J H. Evolution and reuse of orthogonal architecture[J]. IEEE Transactions on Software Engineering, 1996, 22(2): 153-157.

[6] LIU Y P, WU J, LIU S M. Research of Generic Codec for Web Application Testing[J]. Computer Science, 2013, 40(8): 157-160. (in Chinese)
柳永坡，吴际，刘霜梅. Web 应用测试的通用编解码器研究[J]. 计算机科学，2013，40(8)：157-160.

[7] HAN Z W. The Research and Implementation of Universal Decoder Based on TTCN-3[D]. Beijing: Beijing University of Posts and Telecommunications, 2010. (in Chinese)
韩正伟. 基于 TTCN-3 通用解码器的研究与实现[D]. 北京：北京邮电大学，2010.

[8] NING J. The Research and Implementation of Universal Encoder Based on TTCN-3[D]. Beijing: Beijing University of Posts and Telecommunications, 2010. (in Chinese)
宁娇. 基于 TTCN-3 的通用编码器的研究与实现[D]. 北京：北京邮电大学，2010.

[9] SHANG S C. Study on Implementation Technology of Mobile Network Application Software Emulation Test Environment [D]. Beijing: North China University of Technology, 2008. (in Chinese)
尚思超. 手机网络应用软件仿真测试环境开发技术研究[D]. 北京：北方工业大学，2008.

[10] LI W H, MA Y, HUANG X H. Common TTCN-3 Codec to reduce test engineering cost[C]// IET 6th Proceedings of Advanced Intelligence and Awareness Internet, 2010. Ayia Napa: IEEE Press, 2010: 333-336.

[11] ZHANG B. Design and Implementation of TTCN-3 Adapter Library Management System[D]. Beijing: North China University of Technology, 2013. (in Chinese)
张暴. TTCN-3 适配器库管理系统的设计与实现[D]. 北京：北方工业大学，2013.