

基于云计算的 Android 恶意程序协同检测系统

黄兴利¹, 韩艳龙¹, 张长胜², 刘笑言¹, 潘永付¹

(1. 西北工业大学, 陕西 西安 710072;

2. 温州大学, 浙江 温州 325035)

摘要:为检测 Android 恶意程序, 设计了基于云计算的 Android 恶意程序协同检测系统。引入云计算的概念, 通过在云平台上协同调用 N 个杀毒引擎来检测 Android 应用。该系统主要包括两个模块: 信息管理模块和协同检测模块。在检测 Android 应用时, 将应用与特征库进行匹配。若匹配成功, 则显示该应用在特征库中存储的检测结果; 若匹配不成功, 则调用 N 个杀毒引擎来对其进行检测。最终, 系统对 N 个引擎的检测报告进行汇总。利用信誉积分机制获得结果, 将结果反馈并存入系统特征库。实验结果表明, 该系统平均检出率为 97.17%, 检测性能良好且可靠性高。

关键词: 安卓; 恶意程序; 云计算; 信誉积分机制

中图分类号: TP309

文献标识码: A

文章编号: 1673-629X(2016)08-0079-04

doi: 10.3969/j.issn.1673-629X.2016.08.017

An Android Collaborative Malware Detection System Based on Cloud Computing

HUANG Xing-li¹, HAN Yan-long¹, ZHANG Chang-sheng², LIU Xiao-yan¹, PAN Yong-fu¹

(1. Northwestern Polytechnical University, Xi'an 710072, China;

2. Wenzhou University, Wenzhou 325035, China)

Abstract: In order to detect Android malwares, an Android collaborative malware detection system is designed based on cloud computing. Introduction of the concept of cloud computing, Android application is detected through collaboratively calling N antivirus engines in cloud platform. The system mainly consists of information management module and collaborative detection module. The application and characteristics library is mapped when detecting Android applications. If matched, the system will display the result stored in the characteristics library; otherwise, it will call N antivirus engines to detect the application. Eventually, the system summarizes the reports from N antivirus engines. The credit score mechanism is utilized to get the result which is fed back and stored to the characteristics library. The experiments show that the average detection rate of system is 97.17%. Meanwhile, the performance and reliability is high.

Key words: Android; malwares; cloud computing; credit score mechanism

0 引言

近年来, 移动互联网技术发展非常迅速。而 Android 操作系统在全球市场中所占份额远超 50%, 并且还在持续增长。但由于 Android 的开放性, 签名机制的不规范以及应用市场宽松的管理政策, 每个用户都可以编写和发布自己的 Android 应用。这些原因使得 Android 恶意程序的数量与日俱增, 并且可以轻松地逃避杀毒引擎的检测^[1-2]。

因为云计算能够为用户提供大量的信息服务^[3], 所以利用开源项目搭建企业级的云平台成为了一种全

新的计算模式。该模式可以将大量的计算机、虚拟机等网络设备资源整合到云平台下进行管理, 按照一定的商业模式为用户提供服务。因此, 用户可以在没有硬件基础设备的情况下, 访问或操作云平台上共享的应用和数据等信息。这种全新的计算模式是一系列分布式计算发展和演化的结果, 成熟度较高^[4]。

针对如今 Android 恶意程序数量庞大、传播速度快、恶意代码家族众多等特点, 文中提出一种基于云计算的 Android 恶意程序协同检测系统。该系统能够给用户一个最直观的检测结果, 并且该结果具有非常高的准确性。

收稿日期: 2015-11-16

修回日期: 2016-03-09

网络出版时间: 2016-08-01

基金项目: 浙江省自然科学基金(LY14F020030); 公益性技术应用研究计划(2014C31079)

作者简介: 黄兴利(1981-), 男, 硕士, 研究方向为网络安全; 导师: 慕德俊, 教授, 研究方向为信息安全。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20160801.0842.016.html>

1 系统设计方案

1.1 设计思路

目前,每一种恶意程序检测方法都有各自的局限性,对不同恶意代码家族而言,检出率存在巨大的差异。文中利用多个杀毒引擎,以此来弥补各个引擎存在的不足之处。系统框架如图1所示。首先由信息管理模块生成应用的唯一标识符(UID),并且与特征库中已存储的唯一标识符进行匹配。如果匹配成功,则表示系统已经检测过该应用。此时,直接显示系统中存储的检测结果;如果匹配失败,那么信息管理模块将Android应用提交到协同检测模块进行检测。随后,协同检测模块调用云平台下的 N 个杀毒引擎进行检测,并根据 N 个杀毒引擎的信誉积分整理出最终检测结果。最后,信息管理模块将根据 N 个杀毒引擎的检测报告和最终的检测结果来更新各引擎的信誉积分,作为下次检测的依据,并且将该应用的唯一标识符和检测结果存储到特征库中。

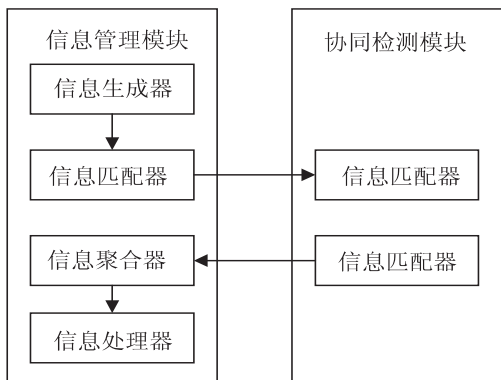


图1 基于云计算的 Android 恶意程序
协同检测系统框架图

1.2 工作流程

检测系统的具体工作流程主要分为以下几个步骤:

- (1) 信息生成器利用哈希算法生成 Android 应用的唯一标识符。
- (2) 信息匹配器将生成的标识符与特征库中进行匹配,如果匹配成功,则直接显示特征库存储的结果。否则,将应用提交给协同检测模块。
- (3) 协同调用器调用 N 个杀毒引擎检测 Android 应用,并将检测结果送到信息聚合器。
- (4) 信息聚合器根据检测杀毒引擎的数量、安全策略、延时等条件,汇总各引擎提供的报告,并将汇总

信息送到信息处理器。

(5) 信息处理器根据信誉积分机制来确定最终的检测结果,同时更新 N 个杀毒引擎的信誉积分。最后,将检测结果存储到特征库中。

图2为检测系统各功能组件工作流程图。

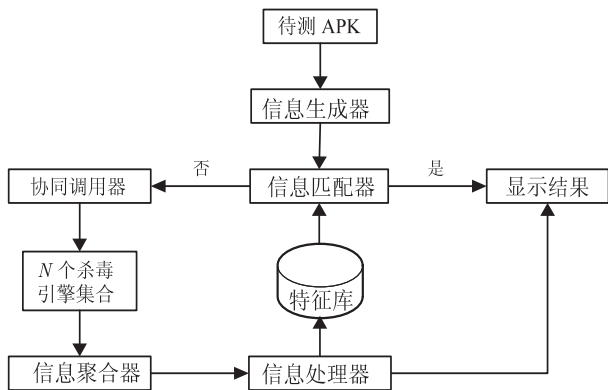


图2 基于云计算的 Android 恶意程序协同检测系统各功能组件工作流程图

2 系统主要组成模块

该系统主要由信息管理模块和协同检测模块构成。

2.1 信息管理模块

信息管理模块是检测系统中的核心模块,主要包括信息生成器、信息匹配器、特征库、信息聚合器、信息处理器。

2.1.1 信息生成器

信息生成器的主要任务是生成应用的标识符 (UID)。由于 Android 应用具有严格的签名机制, 所以生成的 UID 具有唯一性。例如一个应用哈希后生成的 UID, 与该应用重新打包签名后的 UID 是不同的。

2.1.2 信息匹配器

信息匹配器的主要任务是将哈希生成的唯一标识符与特征库进行匹配。如果匹配成功,那么显示特征库中存储的检测结果。否则,将应用提交给协同检测模块。

2.1.3 特征库的建立

面对大量的 Android 应用,需要优化系统,避免不必要的检测。文中通过建立特征库减少冗余的检测时间。特征库的建立有助于优化系统的检测性能和提高系统整体的检测速度。文中的特征库利用 python 中的 MySQLdb 模块^[5]来建立,存储形式如表 1 所示。

表1 特征库存储形式

apkname	apkmd5	time	result
cn.kuwo.player_153401	d2dd4919d84dabfc47655aaf0b68a2a0	2015-9-28	malware
com.achieve.vipshop_182118	3ecfeb0fce1ab9d8d79a759f98e83df3	2015-9-28	benign

2.1.4 信息聚合器

杀毒引擎的检测过程存在明显差异,在协同调用多个引擎时,检测报告的生成时间并不同步,甚至没有反馈报告。虽然 N 个杀毒引擎生成检测报告的时间并不同步,但是信息聚合器可以利用一个子集来代替总集。这里总集代表全部检测引擎的检测报告。文中根据系统部署方案、检测引擎的数量、安全策略和延时等指标选取子集。

信息聚合器中共声明了三个关键参数:数量阈值、收集时间点和终止时间点。从收集时间点开始,如果接收的报告数量大于数量阈值,那么信息处理器将接收这些报告并终止程序;当时间到达终止时间点时,无论生成的报告数量是否大于阈值,这些报告都将被送到信息处理器。

2.1.5 信息处理器

信息处理器对接收到的检测报告进行汇总,并生成最终检测结果。在该阶段,文中利用信誉积分机制确定最终检测结果,同时根据该检测结果与各引擎的检测报告更新 N 个杀毒引擎的信誉积分。最后,将检测结果存储到特征库中。

2.1.6 信誉积分模型

信誉积分代表了杀毒引擎的可信任程度。建立信誉积分机制的目的是根据历史检测信息,预测下次检测的可信任程度。信誉积分机制的提出,有助于提高检测系统的准确率。

文中设计的信誉积分模型^[6-7]的具体思路是:将从信息聚合器发来的报告利用一次加权平均算法,以此确定最终的检测结果。同时,把最终的检测结果和各检测报告进行对比,更新各引擎的信誉积分。

信誉积分机制的模型应遵循以下原则:

(1) 当杀毒引擎的报告存在于检测报告的子集中时,该引擎的信誉积分值才会更新。

(2) 更新杀毒引擎的信誉积分值时,设 R_{init} 为杀毒引擎的初始信誉值, R_x 为更新后的值,而 R_{max} 为信誉的最大值。三者应满足条件 $R_{\text{init}} \leq R_x < R_{\text{max}}$, R_x 在 R_{init} 和 R_{max} 之间改变。在该模型中,文中设 $R_{\text{init}} = 50\%$, $R_{\text{max}} = 100\%$ 。

信誉积分模型算法包括以下几个方面:

(1) 加权平均决策法。

文中使用决策的方法对 N 个杀毒引擎^[8-9]的检测结果进行最终的识别判断。加权平均决策法就是在不确定的情况下,凭借经验对各个方案实施后的情况进行加权平均择优,它能够克服乐观决策法和悲观决策法的缺点。

将加权平均决策法应用到信誉积分模型中就是将每个杀毒引擎的信誉值作为其对应的权重,结合检测

出的结果进行加权平均计算。

检测出为良性的取 1,检测出为恶意的取 0,用 F 表示整个向量集 $F = \{f_1, f_2, \dots, f_x\}$;用 W 表示 N 个杀毒引擎的信誉值向量集 $W = \{R_1, R_2, \dots, R_x\}$ 。用加权平均决策计算表达式为:

$$\bar{x} = \sum_{i=1}^x \frac{f_i R_i}{R_i} = \frac{f_1 R_1 + f_2 R_2 + \dots + f_x R_x}{R_1 + R_2 + \dots + R_x} \quad (1)$$

如果 $\bar{x} \geq 0.5$,则裁定最终检测结果为良性;如果 $\bar{x} < 0.5$,则为恶意。

(2) 权值更新规则。

令原信誉值为 R_{x-1} ,新信誉值为 R_x 。将加权平均决策算法判定出的最终检测结果与 N 个杀毒引擎的检测结果进行对比。若检测结果一致,便可以采用式(2)进行计算:

$$R_x = (1 - R_{x-1})/n + R_{x-1} \quad (2)$$

若干次计算后 R_x 无限逼近 100%,当达到 99% 之后变化不明显。对 n 的选取上要考虑经过多少次达到 99%,次数太少中间数据变化幅度大欠缺准确性。文中采用试差法确定参数 n 的值,经过多次数据计算得出次数 K 与 n 的关系为: $K = 2 + 4(n - 1)$ 。当 n 取 8 时,在第 30 次的变化幅度上接近 0.001,基本上已经达到一个合理的变化范围。因此 n 选取 8,即:

检测结果一致的,采用式(3):

$$R_x = (1 - R_{x-1})/8 + R_{x-1} \quad (3)$$

检测结果不一致的,采用式(4):

$$R_x = R_{x-1} - \left(R_{x-1} - \frac{1}{2}\right)/8 \quad (4)$$

对于未给出检测结果的杀毒引擎信誉值降低的幅度比结果显示不一致的杀毒引擎的信誉值降低的幅度要多。

考虑到降低的幅度过小不如不检测,但要给其留有一定的回升机会,降低的幅度亦不能过大。因此 n 选取 4,既可以拉开层次,同时对信誉值的变化起到了保护作用。

对未给出检测结果的,采用式(5):

$$R_x = R_{x-1} - \left(R_{x-1} - \frac{1}{2}\right)/4 \quad (5)$$

2.2 协同检测模块

协同检测模块的主要任务是调用 N 个杀毒引擎分析检测 Android 应用,主要包括协同调用器和 N 个杀毒引擎集合。协同调用器管理 N 个杀毒引擎,而 N 个杀毒引擎则负责相关计算。

文中的协同检测模块利用 OpenStack^[10-11]技术来实现云计算平台的部署。核心思想是利用虚拟化环境,在一台物理主机上部署虚拟出的多个虚拟机来共同完成多项任务。同时,在 OpenStack 部署好的云环境下创建虚拟机并管理虚拟机。

3 实验与结果

3.1 实验数据

从 2014 年 3 月 7 日到 2014 年 6 月 30 日,利用网络爬虫^[12-14]程序从 Android Malware Genome Project (北卡)、Droid Analytics(香港中文大学)和 virusshare.com 网站上爬取了 27 593 个 Android 应用程序样本作为性能评估数据。同时,选取 2 600 个已知样本来验证系统准确率,其中训练集 600 个,测试集 2 000 个。

3.2 训练

选取 10 个杀毒引擎作为训练工具,利用训练集进行训练。随后,对训练情况进行总体的汇总,见表 2。

表 2 杀毒引擎训练情况汇总表

类别	检测结果						
	检测情况			对比情况		检测后	
	良性	恶意	未反馈	正确	错误	未反馈	信誉值/%
金山毒霸	204	267	129	402	69	129	84.37
江民杀毒	187	280	133	387	80	133	79.20
卡巴斯基	193	326	81	452	67	81	78.94
趋势科技	201	362	37	453	110	37	72.97
ClamAV	156	278	166	348	86	166	68.16
Dr. Web	235	250	115	346	139	115	82.48
360	261	299	40	394	166	40	76.04
Microsoft	321	205	74	426	100	74	78.54
AVG	270	242	88	394	118	88	88.64
McAfee	187	298	115	356	129	115	87.46

经训练可知,系统在检测时能够充分考虑当前杀毒引擎的可信程度以及最近几次检测时的状态,而最终检测结果的确定取决于杀毒引擎集合的决策以及各自信誉值的实际情况。同时,在 10 个杀毒引擎中,错误的检测结果所占的比例最高达到了 27.66%,表明单一引擎在检测恶意程序时存在明显不足。

3.3 测试

将测试集分为 10 组,测试共持续 30 天,系统的平均检出率为 97.17%。从图 3 中可以看出,系统最后趋于稳定,其检出率渐渐趋近于 97.2%,具有非常高的准确率。

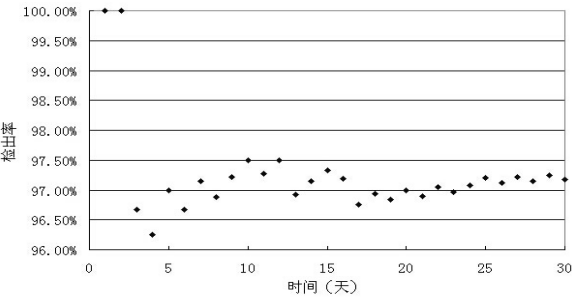


图 3 系统测试检出率散点图

4 结束语

文中通过分析当前 Android 所面临的安全威胁,结合国内外的研究现状,以及分析现有的杀毒引擎综合测试情况,设计并实现了基于云计算的 Android 恶意程序协同检测系统。通过对协同检测系统的相关性能进行训练和测试,证明了文中所提出的模型、方法及相关算法的正确性。实验中训练和测试的检测结果体现了该方法的优势,基于协同的检测方法具有极高的准确率。

参考文献:

[1] 曾健平,邵艳洁. Android 系统架构及应用程序开发研究[J]. 微计算机信息,2011,27(9):1-3.

[2] 李 佳. Android 平台恶意软件检测评估技术研究[D]. 北京:北京邮电大学,2012.

[3] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. Communications of the ACM, 2010, 53(4):50-58.

[4] Wang L, von Laszewski G, Younge A, et al. Cloud computing: a perspective study[J]. New Generation Computing, 2010, 28(2):137-146.

[5] Dustman A. MySQLdb: a Python interface for MySQL[EB/OL]. 2010. <http://mysqldbpython.sourceforge.net/MySQLdb.html>.

[6] Huynh T D, Jennings N R, Shadbolt N R. An integrated trust and reputation model for open multi-agent systems[J]. Autonomous Agents and Multi-agent Systems, 2006, 13(2):119-154.

[7] 辛海涛. P2P 网络中信任模型的研究[D]. 武汉:武汉理工大学,2011.

[8] Oberheide J, Cooke E, Jahanian F. CloudAV: N-version anti-virus in the network cloud[C]//Proc of USENIX security symposium. [s. l.]:[s. n.], 2008:91-106.

[9] 关卫中,秦 攀,邹德清,等. 云防御系统中多引擎检测机制[J]. 武汉大学学报:理学版,2014,60(5):393-398.

[10] Pepple K. Deploying OpenStack[M]. [s. l.]:O'Reilly Media, Inc, 2011.

[11] Sefraoui O, Aissaoui M, Eleuldj M. OpenStack: toward an open-source solution for cloud computing[J]. International Journal of Computer Applications, 2012, 55(3):38-42.

[12] 周德懋,李舟军. 高性能网络爬虫:研究综述[J]. 计算机科学, 2009, 36(8):26-29.

[13] 李志义. 网络爬虫的优化策略探略[J]. 现代情报, 2011, 31(10):31-35.

[14] 赵茉莉. 网络爬虫系统的研究与实现[D]. 成都:电子科技大学, 2013.