Computer Science and Application 计算机科学与应用, 2017, 7(6), 491-498
Published Online June 2017 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2017.76060

Hans汉斯

A Malware Detection Model Based on Behavior Analysis and KNN Algorithm

Chunbo Ma, Kun Zeng

School of Information and Communication, Guilin University of Electronic Technology, Guilin Guangxi
Email: machunbo@guet.edu.cn, 2313574812@qq.com

Received: May 28th, 2017; accepted: Jun. 11th, 2017; published: Jun. 14th, 2017

Abstract

In order to solve the problem that traditional malware detection technology can't deal with unknown malware, a malware detection model which is based on behavior analysis and KNN algorithm is proposed. On the basis of summarized behavior characteristics of malware, the model adopts information gain method to reduce dimensionality of behavior characteristics, and utilizes the behavior extraction engine which is based on open source sandbox to analyze and extract behavior characteristics and uses improved KNN algorithm in the detection engine to detect malware. The results of experiment demonstrate that the model has a good capability to detect unknown malware, and achieves a high FPR, AR and a low FPR.

Keywords

Malware, Behavior Characteristic, KNN Algorithm, Detection Model

一种基于行为分析和KNN算法的恶意软件检测模型

马春波，曾　坤

桂林电子科技大学信息与通信学院，广西 桂林
Email: machunbo@guet.edu.cn, 2313574812@qq.com

收稿日期：2017年5月28日；录用日期：2017年6月11日；发布日期：2017年6月14日

摘　要

为了解决传统恶意软件检测技术无法应对未知恶意软件的问题，提出了一种基于行为分析和KNN算法的

文章引用：马春波，曾坤．一种基于行为分析和 KNN 算法的恶意软件检测模型[J]．计算机科学与应用，2017，7(6): 491-498. https://doi.org/10.12677/csa.2017.76060

恶意软件检测模型。该模型以总结归纳的恶意软件行为特征为基础，采用信息增益算法对行为特征进行降维，利用基于开源沙盒的行为提取引擎对样本的行为特征进行分析和提取，并在检测引擎中使用改进的KNN算法进行恶意软件的检测。实验结果表明，此模型具有良好的未知恶意软件检测能力，同时实现了较高的检测率、准确率以及较低的误报率。

关键词

恶意软件，行为特征，KNN算法，检测模型

1. 引言

随着计算机技术和 Internet 的高速发展，暴露在网络中的计算机面临着越来越多的恶意软件威胁。近年来，伴随着黑客技术的低门槛化以及经济利益的驱动，恶意软件的产生呈现出规模化、专业化等特点，更有甚者形成了一条地下黑色产业链，其数量、破坏性以及反查杀能力都得到了极大的增长[1]。恶意软件的攻击会给个人和团体造成难以挽回的损失，这凸显了反恶意软件技术的重要性。恶意软件的检测是反恶意软件技术中的重要环节，近年来已成为信息安全领域的研究热点之一。

传统的恶意软件检测方法是建立在静态分析的基础上[2]。所谓静态分析，就是在不执行未知软件样本的前提下，将样本的特征码提取出来，并与恶意软件特征码库中的数据进行匹配来判断未知样本的恶意与否。该方法具有检测准确、误警率低等特点[3]。但由于一方面需要事先建立大量的恶意软件特征码库作为匹配的依据；另一方面，由于加壳、变形、隐藏等反查杀技术的出现，不仅使恶意软件的特征码极易发生改变，也让特征码的提取工作变得困难重重，这导致了该项技术的检测能力远远落后于恶意软件的肆意扩张。尤其是在面对新型计算机病毒、木马等程序时，静态分析法常常显得束手无策，这就对恶意软件的检测技术提出了新的要求。

基于行为分析的检测是一种动态分析技术，可以很好的弥补静态分析法在这方面的不足。由于恶意软件的行为与正常软件的行为具有很大的区别，通过对样本运行过程中的行为进行捕捉，分析和提取出样本的行为特征，以此作为样本检测和判别的依据。在行为检测的基础上进一步采用虚拟机与沙盒技术，让软件样本在虚拟的计算机环境中执行，不但可以捕捉到软件样本的行为记录，还可以避免恶意软件对真实的计算机系统产生破坏，从而达到我们保护计算机安全的目的。

本文以此为依据，提出了一种基于行为分析和 KNN 算法的恶意软件检测模型。该模型首先对恶意软件的行为特征进行归纳总结，采用信息增益算法对行为特征进行降维，并将其使用于行为提取服务器中；然后采用基于开源沙盒的行为提取引擎对样本的行为特征进行分析提取和数据库存储；最后以行为特征为依据，在检测服务器中使用改进的 KNN 算法进行恶意软件检测。实验结果表明，本模型能够准确地对恶意软件进行检测与识别。

2. 检测模型结构

2.1. 整体框架结构

检测模型的整体框架结构如图 1 所示，主要包括行为提取引擎、检测引擎、行为特征数据库以及其

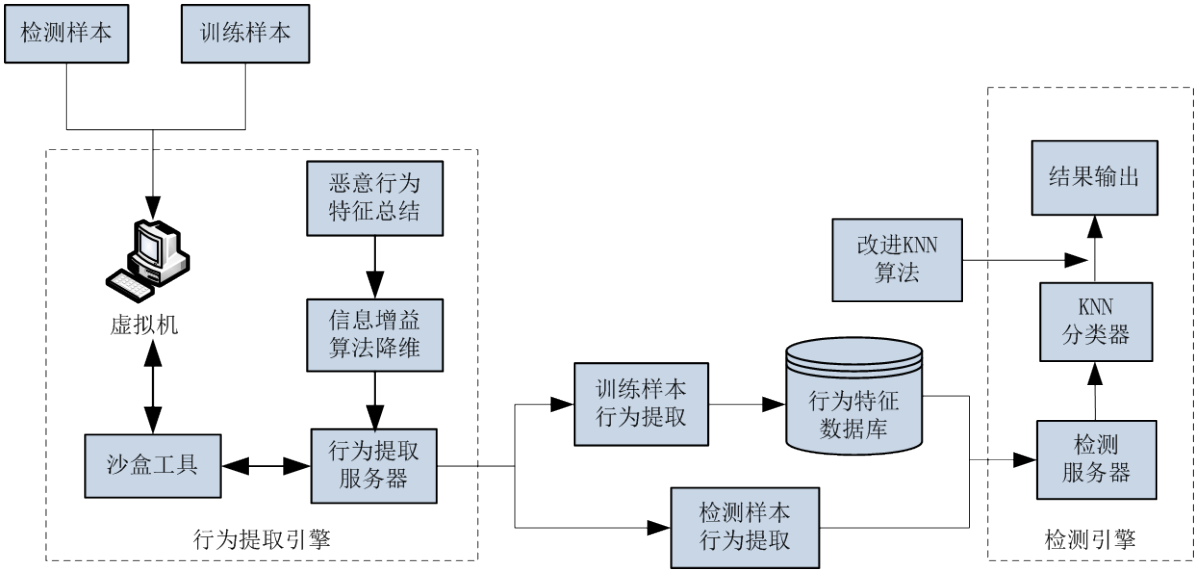

Figure 1. General frame of detection model
图 1. 检测模型整体框架

它几个部分。

1) 行为特征是根据大量实验和专家经验总结而来，除此之外还要利用信息增益理论对行为特征进行降维以提高检测的准确度和降低开销，最终选取出的行为特征将作为行为提取服务器的分析提取依据。

2) 行为提取引擎主要由行为提取服务器以及沙盒工具组成，负责对训练和检测样本进行行为分析和提取。样本在虚拟环境中运行，可以防止其对真实的计算机系统造成损害，同时利用沙盒工具的监控脚本对软件的运行过程进行监控，沙盒工具将分析结果(报告文件)输出到行为提取服务器上。

3) 对于训练样本而言，其行为提取结果会被放入行为特征数据库中，用于训练 KNN 分类器模型(二分类)。对于检测样本而言，其行为特征成为了检测的重要依据，最终将被输入检测引擎中进行检测样本的识别。

4) 检测引擎主要包括检测服务器和 KNN 分类器等部分。检测服务器负责将输入数据转交给分类器，分类器接受行为特征数据库和检测样本行为特征的输入，利用改进的 KNN 算法进行最后的检测，并得到最终判别结果。

2.2. 行为特征的确定

行为特征确定的方法如下：

1) 首先收集大量的恶意软件样本，在虚拟环境中让恶意软件得到充分地运行，并对其动态行为进行监控。利用常规知识和经验分辨出与正常软件区分性较大的行为，记录下来并进行汇总，以此作为恶意软件的行为特征。

2) 如果仅仅是利用上述方法，可能会出现遗漏和错误的情况，因此还有很重要的一点就是翻阅文献资料，同时结合信息安全领域专家的经验，共同确定恶意软件通用的行为特征[4]。

2.3. 基于信息增益的行为特征降维

由于恶意软件行为特征的维数很大，因此会导致检测的计算复杂度增加，所以必须对于这些行为特征进行有效性筛选。

将恶意软件检测模型视为一个系统，那么对于这个系统而言，其类别变量集合表示为$C=\{c_1,c_2\}$，其中c_1表示恶意软件类别，c_2表示正常软件类别。系统的熵定义为

$$H(C)=-\sum_{i=1}^{2}P(c_i)\log_2 P(c_i) \tag{1}$$

假设行为特征向量表示为T，考虑到如果该系统中某行为特征T_i没有出现，或者T_i被确定下来了，那么此时的系统熵被表示为该行为特征对系统的条件熵，也就是：

$$H(C|T)=\sum_{i=1}^{n}P(T_i)H(C|T=T_i) \tag{2}$$

其中$P(T_i)$表示特征向量$T=T_i$的概率。

信息量期望值或者说信息熵的减少量用信息增益*IG*来表示，它反映了某项行为特征的出现对整个系统熵值的影响，或者说是为整个系统带来了多少信息量[5]。对于某项行为特征，其信息增益可以用该特征确定前的系统熵与确定后的条件熵之差来表示，表达公式如下：

$$\begin{aligned} IG(T)&=H(C)-H(C|T)\\ &=-\sum_{i=1}^{n}P(c_i)\log_2 P(c_i)+\sum_{j=1}^{m}\left[\sum_{i=1}^{n}P(c_i|T_i)\log_2 P(c_i|T_j)\right] \end{aligned} \tag{3}$$

根据信息增益的公式可知，某一行为特征相对系统的条件熵越小，则最后的信息增益值越大。因此只需要计算每个行为特征相对于系统的条件熵，然后按照“条件熵大信息增益小”的原则[6]，将所有行为特征的条件熵按照由小到大的顺序进行有序排列，即得到行为特征的排列序列。门限阈值的确定根据检测模型的实际要求来定，最后依照阈值对恶意软件的行为特征进行筛选，实现基于信息增益的行为特征降维。

最终经过降维的恶意软件行为特征及完成相应行为需要用到的API函数如表1所示。

2.4. 开源沙盒工具 Cuckoo Sandbox

Cuckoo Sandbox是一款基于GPLv3开源协议的沙盒工具，可以用于对恶意软件进行初步分析，其主要功能包括：

1) 追踪恶意代码进程的调用情况；
2) 监测恶意代码执行过程中删除、下载或者新建的文件；
3) 以PCAP格式保存恶意代码的网络通信；
4) 获取恶意代码的内存镜像；

Cuckoo Sandbox的结构如图2所示，主要包括Host Machine和Guest Machine两大部分，Host Machine和Guest Machine之间通过虚拟网络进行通信。

Host Machine即中央管理模块，包括Cuckoo Sandbox本体、各类分析组件以及虚拟机软件Virtual Box，主要负责启动分析，行为监控以及报告文件的产生等等。

Guest Machine即虚拟机环境，主要负责执行恶意代码并向中央管理模块汇报分析结果，值得注意的是，Cuckoo Sandbox在Guest Machine中注入的有一个监控脚本。

Cuckoo Sandbox的工作流程为：

1) 首先Cuckoo Sandbox执行主程序脚本，同时开启Guest Machine虚拟机。
2) 通过上传功能脚本和监控脚本，将样本送入Guest Machine中，准备执行。
3) 样本在Guest Machine中执行，同时监控脚本对样本的各种信息进行记录。

Table 1. Behavior characteristics of malware
表 1. 恶意软件行为特征

行为特征类别	行为特征描述	调用 API
文件行为	创建文件	CreateFile
	读取文件	ReadFile
	写入文件	WriteFile
	删除文件	DeleteFile
	复制文件	MoveFile
	移动文件	CopyFile
	更改文件属性	SetFileAttributes
磁盘行为	获取系统路径	GetSystemDirectory
	磁盘剩余空间	GetDiskFreeSpace
	获取磁盘类型	GetDriveType
	遍历磁盘文件	FindFirstFile, FindNextFile
网络行为	打开网络链接	InternetOpenUrl
	远程下载文件	UrlDownloadToFile
	从远程主机接收数据	Accept, Recv
	远程发送数据	Send
服务行为	连接服务管理器	OpenSCManager
	打开服务	OpenService
	枚举服务	EnumServicesStatus
	检索服务	QueryServiceConfig
	控制服务	ControlService
注册表行为	创建注册表项	RegCreateKey
	修改注册表项	RegSetValue
	查找注册表项	RegQueryValue
	删除注册表项	RegDeleteKey, RegDeleteValue
进程行为	创建进程	CreateProcess
	加载动态库	LoadLibrary
	获取进程信息	GetCurrentProcess, GetPriorityClass
	关闭或释放进程	ExitProcess, TerminateProcess

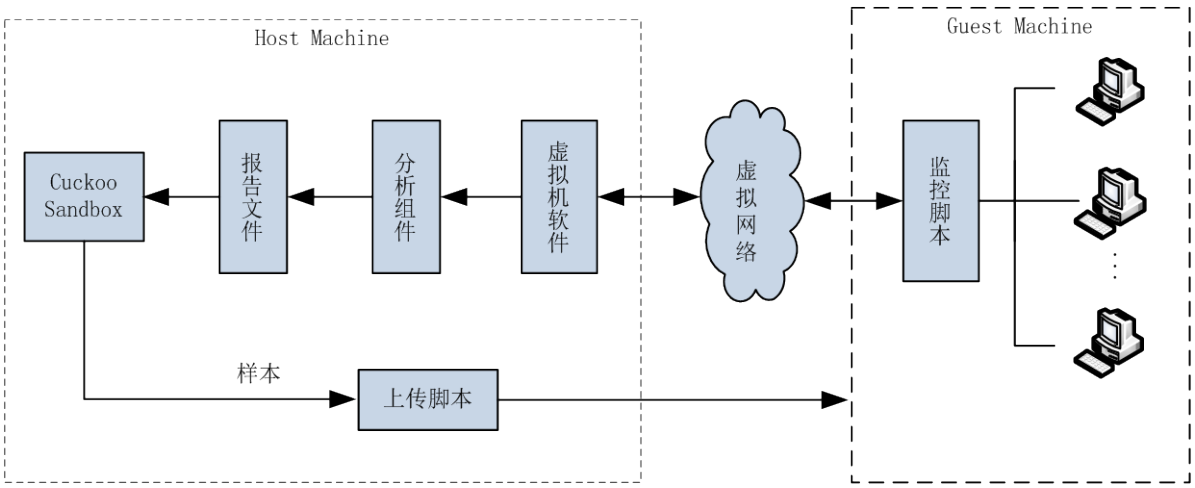

Figure 2. Structure of cuckoo sandbox
图 2. Cuckoo sandbox 结构

4) 样本执行完毕，监控脚本通过虚拟网络以及虚拟机软件的共享功能将记录发送给虚拟机软件所在的 Host Machine。

5) 虚拟机软件利用快照将 Guest Machine 虚拟机恢复到初始状态。

6) 记录结果通过分析组件生成报告文件。

报告文件中包含了恶意代码的行为信息，至此，恶意代码的行为通过行为提取引擎被成功地提取出来。

2.5. 基于行为特征的 KNN 检测算法

假定样本集合 S 被划分为训练样本集合和测试样本集合。训练样本集合表示为 $S_N=\{N_1,N_2,\cdots,N_p\}$，测试样本集合表示为 $S_T=\{T_1,T_2,\cdots,T_q\}$，样本类别集合 $C=\{c_1,c_2\}$。

假设经过降维后的行为特征，其维数为 n。那么对于 S 中的每个样本 x 而言，将其投射到 n 维特征空间中用行为特征向量表示，即 $x=[a_1(x),a_2(x),\cdots,a_n(x)]$。

KNN 算法基于一定的假设前提[7]，即特征空间中某样本的类别与其附近的其它样本类别更为相似，且样本间的距离越小，其相似程度越高。对于 n 维特征空间中的实例而言，将两个样本实例之间的距离度量表示为欧氏距离 $d(x_i,x_j)$

$$d(x_i,x_j)=\sqrt{\sum_{k=1}^{n}\left[a_k(x_i)-a_k(x_j)\right]^2} \tag{4}$$

对于测试样本集合 S_T 中的任一样本 x_T，计算出其与 q 个训练样本的距离，用集合表示为 $D=\{d(x_T,N_1),d(x_T,N_2),\cdots,d(x_T,N_q)\}$，选取其中距离最小的 k 个训练样本用于测试样本的类别判断。

假定这 k 个近邻样本中，属于 c_i 类的样本数量为 k_i (其中 i 的取值为 1 或 2)，判别函数表达式为 $g_i(x_T)=k_i$，则最终决定测试样本分类的决策函数表达式为

$$\text{若 } g_j(x_T)=\max k_i\text{，则 } x_T\in c_i \tag{5}$$

2.6. KNN 检测算法的改进

对于基于行为特征的 KNN 检测算法而言，不同的行为特征对于检测结果的贡献值是存在着差别的，有的行为特征可以起到决定性作用，有的行为特征起次要作用。

然而从式(4)可以看出样本的各项行为特征对于样本之间的距离的影响是均等的，这对恶意软件的检测并无好处，因为各项行为特征对应的可疑级别并不相同。

为了使 $d(x_i,x_j)$ 能更准确地反映出样本之间的距离，我们根据恶意软件行为特征的特点对各项行为特征 a_k 授予不同的权值 ω_k。由此样本间的距离度量公式改为

$$d(x_i,x_j)=\sqrt{\sum_{k=1}^{n}\omega_k\left[a_k(x_i)-a_k(x_j)\right]^2} \tag{6}$$

如果测试样本 x_T 的周围的样本点的个数较少，那么该 k 个点所覆盖的区域将会很大，反之则小。因此最近邻算法易受样本空间中孤立点的影响。其根源在于基本的 KNN 算法中，测试样本 x_T 的 k 个最近邻样本的地位是平等的。

在自然社会中，一个对象受其不同近邻的影响是不同的，通常是距离越近的对象对其影响越大[8]。在实际应用中我们也发现，测试样本 x_T 距它的 k 个近邻距离不同，对决策的贡献也不同。样本之间不同的距离拥有不同的贡献度，距离越近的样本越相似。所以我们用近邻的距离加权 μ_r 来体现近邻的贡献程度。这里取样本间距离度量平方的倒数为其距离的权值，即

$$\mu_r = \frac{1}{d\left(x_i, x_j\right)^2} \tag{7}$$

由此，决策规则改进为二维判定，不仅要比较不同类别中样本的个数，而且要比较不同种类的 r 个样本距离加权的和。最终改进后的决策函数表达式为：

$$\text{若 } g_1\left(x_T\right) = \begin{cases} r_1 > r_2 \\ \sum_{i \in c_1} \mu_i d\left(x_T, x_i\right) > \sum_{i \in c_2} \mu_j d\left(x_T, x_j\right) \end{cases}, \text{ 则 } x_T \in c_1, \text{ 否则 } x_T \in c_2 \tag{8}$$

3. 实验结果与分析

首先利用收集的 927 个正常软件和 2352 个恶意软件建立实验样本库，正常软件组成正常样本集，恶意软件组成恶意样本集，如表 2 所示。

然后在正常样本集和恶意样本集共同组成的样本库中，随机选取一定比例的样本作为训练样本集，其余样本作为测试样本集。利用检测模型对测试样本进行检测，根据结果的正确与否计算出实验结果数据。

最后重复 10 次实验，最终的实验结果取为 10 次实验结果的平均值，以确保结果更为精确。

本文选取文献[9]、文献[10]、文献[11]中的检测结果作为对比，得到的实验结果如表 3 所示，数据单位为百分比。实验结果中的 TPR 表示恶意软件被正确检测出的概率即检测率，FPR 表示正常软件被误报为恶意软件的概率即误报率，AR 表示检测正确的综合概率即准确率。

Table 2. Samples of experiment
表 2. 实验样本

样本类别	数量
正常样本	927
恶意样本	2352

Table 3. Results of malware detection experiment
表 3. 恶意软件检测实验结果

结果序号	本文模型			文献[9]模型		文献[10]模型		文献[11]模型	
	AR	TPR	FPR	AR	FPR	TPR	FPR	TPR	AR
1	97.1	96.2	4.3	94.8	6.0	93.59	9.92	93.0	95.1
2	92.9	92.7	7.5	93.2	5.8	91.68	9.48	74.0	98.6
3	96.5	96.9	4.1	94.4	5.7	81.21	19.48	65.0	80.0
4	95.3	95.0	5.9	93.8	5.6	92.98	9.19	95.0	95.8
5	98.0	97.8	3.4	94.2	5.5	93.63	10.28	77.0	99.1
6	96.4	95.9	4.5	-	5.5	92.27	9.69	64.0	99.4
7	97.6	97.3	4.1	-	5.6	92.31	9.15	93.0	98.6
8	97.2	97.0	3.2	-	5.5	-	-	72.0	98.4
9	95.1	94.5	5.1	-	-	-	-	61.0	99.7
10	95.5	95.8	4.8	-	-	-	-	-	-
平均值	96.16	95.91	4.69	94.08	5.65	91.09	11.02	77.11	96.07

在检测率方面，本文(95.91%)与文献[10] (91.09%)设计的检测模型都达到了很高的水平，保持在 90%以上，其中前者检测率高出后者 4.82%；而文献[11] (77.11%)设计的 MCPMVE 模型由于受恶意软件占比和查询深度影响，会出现一定幅度的波动，导致其最终的检测结果和 TPR 平均值不理想。

准确率方面，本文(96.16%)与文献[9] (94.08%)、文献[11] (96.07%)的检测系统均达到了很高的水平，三者不相上下，其中本系统放入检测准确率相比其它二者略微占优。

误报率方面，本文(4.69%)与文献[9] (5.65%)的系统误报率均维持在 5%左右较低的水平，其中前者误报率略低于后者；而文献[10]的误报率为 11.02%，是前两者的 2 倍左右。

检测实验结果表明本系统具有很好的恶意软件检测识别能力，与选取的同类型检测系统相比具有更高的检测率与准确率，同时实现了更低的误报率。

4. 结束语

本文针对传统恶意软件检测方法存在的不足，提出基于行为分析和 KNN 算法的检测模型。该模型以总结归纳的恶意软件行为特征为依据，将其降维后运用于行为提取引擎中，并在检测引擎中采用改进后的 KNN 算法进行恶意软件的检测。实验结果表明，此模型对于未知恶意软件亦具有良好的检测识别能力，同时实现了较高的检测率、准确率以及较低的误报率。后续工作将往更多平台以及更多文件类型方面进行尝试，以及进一步研究恶意行为特征的选取降维以及检测算法的改进。

资助信息

受广西精密导航技术与应用重点实验室资助(DH201503)。

参考文献 (References)

[1] Rieck, K., Trinius, P. and Willems, C. (2011) Automatic Analysis of Malware Behavior Using Machine Learning. *Journal of Computer Security*, **19**, 639-668. https://doi.org/10.3233/JCS-2010-0410

[2] Ding, Y., Dai, W. and Yan, S. (2014) Control Flow-Based Opcode Behavior Analysis for Malware Detection. *Computers and Security*, **44**, 65-74. https://doi.org/10.1016/j.cose.2014.04.003

[3] 吴冰, 云晓春, 高琪. 基于网络的恶意代码检测技术[J]. 通信学报, 2007, 28(11): 92-96.

[4] Hisham, S.G., Yousef, B.M. and Mohammed, A.A. (2016) Behavior-Based Features Model for Malware Detection. *Journal of Computer Virology and Hacking Techniques*, **12**, 59-67. https://doi.org/10.1007/s11416-015-0244-0

[5] Engin, K. (2006) Behavior Based Spyware Detection. *Proceedings of the* 15*th USENIX Security Symposium*, Vancouver, 5-9 August 2002, 246-253.

[6] 张小康, 帅建梅, 史林. 基于加权信息增益的恶意代码检测方法[J]. 计算机工程, 2010, 36(6): 149-151.

[7] 赵云程, 慕德俊, 戴航. 双重恶意代码检测系统的设计与实现[J]. 计算机技术与发展, 2013, 23(10): 111-114.

[8] 韩小素, 庞建民, 岳峰. 一种恶意代码变种检测的有效方法[J]. 计算机安全, 2010(9): 53-57.

[9] 张程. 基于行为检测的恶意代码查杀引擎技术研究[D]: [硕士学位论文]. 北京: 北京邮电大学, 2012.

[10] 赵恒立. 恶意代码检测与分类技术研究[D]: [硕士学位论文]. 杭州: 杭州电子科技大学, 2009.

[11] 龚培娇. 虚拟环境下恶意代码检测技术与防范模型的研究[D]: [硕士学位论文]. 西安: 西安建筑科技大学, 2014.