文本复制检测报告单(去除本人已发表文献)

№:ADBD2017R_201712191108082017121915514712035290770 11　　检测时间:2017-12-19 15:51:47

检测文献:　2015110856_朱鹏博_基于机器学习算法的恶意代码检测技术研究-预审

作者:　朱鹏博

检测范围:　中国学术期刊网络出版总库

中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库

中国重要会议论文全文数据库

中国重要报纸全文数据库

中国专利全文数据库

互联网资源(包含贴吧等论坛资源)

英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)

港澳台学术文献库

优先出版文献库

互联网文档资源

图书资源

CNKI大成编客-原创作品库

学术论文联合比对库

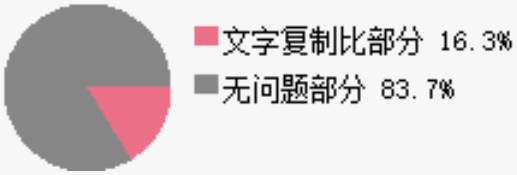
个人比对库

时间范围:　1900-01-01至2017-12-19

检测结果

去除本人文献文字复制比：16.3%

重复字数：	[5710]	总段落数：	[7]
总字数：	[34924]	疑似段落数：	[6]
疑似段落最大重合字数：	[2857]	前部重合字数：	[437]
疑似段落最小重合字数：	[33]	后部重合字数：	[5273]



指　标：☑疑似剽窃观点　☑疑似剽窃文字表述　☐疑似自我剽窃　☐疑似整体剽窃　☐过度引用

表格：0　　脚注与尾注：0

6.7%（332）　中英文摘要等（总4933字）

21.2%（1293）　第一章绪论（总6111字）

44%（2857）　第二章相关理论与关键技术（总6497字）

10.7%（1052）　第三章基于机器学习算法的恶意代码检测方法_第1部分（总9790字）

0%（0）　第三章基于机器学习算法的恶意代码检测方法_第2部分（总3546字）

1.2%（33）　第四章系统设计与实现（总2834字）

11.8%（143）　第五章总结与展望（总1213字）

（注释：无问题部分　文字复制比部分）

疑似剽窃观点　(1)

第二章相关理论与关键技术

1. 需要利用恶意代码分析技术和监控技术来获取恶意代码的基本属性和执行信息，进一步了解恶意代码的功能，实现恶意代码的检测并抑制恶意代码。

1. 中英文摘要等　　总字数：4933

相似文献列表　文字复制比：6.7%(332)　疑似剽窃观点：(0)

1	10S051019-卢占军-丁宇新 卢占军 -《学术论文联合比对库》- 2012-12-12	4.2%（207） 是否引证：否

2	BH2009921397	4.2%（207）
	-《学术论文联合比对库》- 2012-12-04	是否引证：否
3	BH2009925453	4.2%（207）
	-《学术论文联合比对库》- 2012-12-11	是否引证：否
4	基于操作码序列的静态恶意代码检测方法的研究	3.4%（168）
	卢占军(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2012-12-01	是否引证：否
5	11S051036-陈晨-丁宇新	2.1%（106）
	陈晨 -《学术论文联合比对库》- 2013-11-27	是否引证：否
6	11S051036-陈晨-丁宇新	2.1%（106）
	陈晨 -《学术论文联合比对库》- 2013-11-14	是否引证：否
7	基于程序语义的静态恶意代码检测系统的研究与实现	0.9%（43）
	袁雪冰(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2009-12-01	是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

密级： 保密期限：

硕士学位论文

题目：基于机器学习算法的恶意代码检测技术研究

学号：

姓名：

专业： 计算机科学与技术

导师：

学院： 计算机学院

2017年 12月 15日

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名： 日期：

关于论文使用授权的说明

本人完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。

本学位论文不属于保密范围，适用本授权书。

本人签名： 日期：

导师签名： 日期：

基于机器学习算法的恶意代码检测技术研究

摘要

飞速发展的互联网技术推动人类社会的不断进步，但凡事皆有利弊，互联网技术的不断革新也促进了恶意代码的发展壮大，其危害也变得越来越难以控制。在如今信息高度共享的时代，计算机信息安全同样面临着巨大的挑战，各种网络安全事件频发，给个人、企业甚至政府机关带来了巨大经济损失。在众多的攻击事件中，恶意代码是其中的主要的攻击手段。

随着信息共享技术的发展及使用，恶意代码的发展速度和更新频率变得越来越快，各种未知恶意代码层出不穷，给分析人员带来巨大的挑战。一般来说，恶意代码分析技术分为静态分析和动态分析，静态分析技术从语法、语义的层面去理解程序的行为，以期望获取程序在运行过程中的信息，而不需要运行程序。动态分析技术指在可控环境下实际运行程序，监控执行过程中的程序行为，记录程序执行的信息。传统的基于特征匹配的检测方法对于各种变形的恶意代码显得无能为力。随着机器学习和数据挖掘技术的快速发展，已经有许多研究将这些技术用于恶意代码的实际检测环境中，并且取的了不错的效果。如何应对恶意代码的爆发式增长，尤其是对未知恶意代码及其变种得到较好的检测效果成为恶意代码检测技术的研究重点。

本文对基于机器学习算法的恶意代码检测方法做出了改进，并根据该方法设计实现了一个恶意代码检测系统。与传统方法不同的是，本文提供多于一种的抽象化方式，并使用Eclat算法对中间码特征序列进行频繁项集分析，根据分析结果在多种中间码特征序列中做出选择，选择对分类效果最为明显的中间码特征序列，并根据此中间码特征序列生成概率矩阵，然后以此概率矩阵作为代表恶意代码的特征。首先，对恶意代码样本进行查壳和脱壳处理；其次，对恶意代码样本进行反汇编处理，得到样本的汇编文本，并从中提取出汇编操作码序列；然后根据定义的多种抽象方式分别对汇编操作码序列进行抽象处理，得到各抽象方式对应的中间码序列；接着，使用n-gram算法获得中间码特征序列，并使用Eclat算法对中间码特征序列的频繁项集进

行分析，预测出对分类效果最明显的中间码特征序列，并依据该中间码特征序列生成概率矩阵作为样本特征；最后，使用机器学习算法实现对恶意代码的检测。在实验中，通过准确率、精确率和召回率三个指标对实验结果进行评估。通过实验结果的对比分析，本文使用的方法相比传统方法有一定的优越性。

关键词：机器学习；恶意代码检测；频繁项集；概率矩阵

MALICIOUS CODE DETECTION TECHNOLOGY BASED ON MACHINE LEARNING ALGORITHMS

ABSTRACT

The rapid development of Internet technology to promote the continuous progress of human society, but every coin has two sides, the continuous innovation of Internet technology also contributed to the development and growth of malicious code. Its harm has become more and more difficult to control. In today's age of highly shared information, computer information security is also faced with huge challenges. All kinds of cyber-security incidents are frequent and have brought huge economic losses to individuals, enterprises and even government agencies. Among the many attacks, malicious code is one of the major attacks.

With the development and use of information sharing technology, the speed of development and updating of malicious code has become faster and faster, and various kinds of unknown malicious code emerge in an endless stream, posing great challenges to analysts. In general, malicious code analysis techniques are divided into static analysis and dynamic analysis. Static analysis techniques understand the behavior of programs from the perspective of grammar and semantics so as to expect to obtain the information during the running process without running programs. Dynamic analysis refers to the actual operation of the program in a controlled environment, monitor the implementation of the program behavior, record the program execution information. The traditional detection methods based on feature matching seem powerless to various malformed code. With the rapid development of machine learning and data mining technologies, many researches have been devoted to using these techniques in the actual detection environment of malicious code, and have achieved good results. How to deal with the explosive growth of malicious code, especially for the detection of unknown malicious code and its variants has become a research focus of malicious code detection technology.

This paper improves the method of malicious code detection based on machine learning algorithm, and designs and implements a malicious code detection system based on this method. Different from the traditional method, this article provides more than one abstraction way, and uses the Eclat algorithm to carry out frequent itemsets analysis of the middle code feature sequence. Based on the analysis result, it makes a choice among a variety of middle code feature sequences, Then the probability matrix is generated according to the middle code feature sequence, and then the probability matrix is taken as the representative of malicious code. Firstly, the malicious code samples are checked and shelled. Secondly, The malicious code samples are disassembled to obtain the assembly texts of the samples and the assembly opcode sequences are extracted therefrom; and then the assembly opcode sequences are abstractly processed according to a variety of abstraction ways defined to obtain the intermediate code sequences corresponding to each abstract mode Then, we use the n-gram algorithm to obtain the middle code feature sequence and use the Eclat algorithm to analyze the frequent item sets of the middle code feature sequence to predict the middle code feature sequence with the most obvious classification effect. Feature sequences generate probabilistic matrices as sample features; finally, machine learning algorithms are used Realize the detection of malicious code In the experiment, through the accuracy, accuracy and recall rate of three indicators to evaluate the experimental results. Through the comparative analysis of the experimental results, the method used in this paper has some advantages over the traditional methods.

KEY WORDS：Machine Learning, Malicious Code Detection, Frequent Itemsets, Probability Matrix

目录

指　标
疑似剽窃文字表述

1. 密级： 保密期限：
   硕士学位论文
   题目：基于机器学习算法的恶意代码检测技术研究
2. 恶意代码及其变种得到较好的检测效果成为恶意代码检测技术的研究重点。
   本文对基于
3. 查壳和脱壳处理；其次，对恶意代码样本进行反汇编处理，得到样本的汇编文本，并从中提取出汇编操作码序列；
4. How to deal with the explosive growth of malicious code, especially
5. research focus of malicious code detection technology.
   This paper

2. 第一章绪论

总字数：6111

相似文献列表　文字复制比：21.2%(1293)　疑似剽窃观点：(0)

1	基于操作码行为深度学习的恶意代码检测方法 陈晨(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2013-12-01	16.1%（985） 是否引证：否
2	11S051036-陈晨-丁宇新 陈晨 -《学术论文联合比对库》- 2013-11-27	15.4%（942） 是否引证：否
3	11S051036-陈晨-丁宇新 陈晨 -《学术论文联合比对库》- 2013-11-14	15.4%（942） 是否引证：否
4	基于操作码序列的静态恶意代码检测方法的研究 卢占军(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2012-12-01	15.0%（918） 是否引证：否
5	10S051019-卢占军-丁宇新 卢占军 -《学术论文联合比对库》- 2012-12-12	13.8%（843） 是否引证：否
6	BH2009925453 -《学术论文联合比对库》- 2012-12-11	13.8%（843） 是否引证：否
7	BH2009921397 -《学术论文联合比对库》- 2012-12-04	13.5%（828） 是否引证：否
8	基于动态污点分析的恶意代码行为依赖图挖掘技术的研究与实现 尤作赛(导师：王勇军) -《国防科学技术大学硕士论文》- 2012-09-01	1.5%（91） 是否引证：否
9	基于脉搏信号的VDT视觉疲劳研究 蔡海燕(导师：张爱华) -《兰州理工大学硕士论文》- 2012-05-10	1.1%（69） 是否引证：否
10	基于改进PSO算法的测试用例生成方法研究 刘瑞(导师：沈夏炯) -《河南大学硕士论文》- 2011-05-01	0.6%（35） 是否引证：否
11	足部三维重构的关键技术研究 田苗苗(导师：杨秋翔;刘磊) -《中北大学硕士论文》- 2013-05-25	0.5%（31） 是否引证：否
12	东洋机电（烟台）有限公司ERP系统的设计与实现 陈晨(导师：王晓琳;巢林) -《山东大学硕士论文》- 2010-04-05	0.5%（29） 是否引证：否
13	可控僵尸网络模拟平台的研究与实现 丁澄天(导师：刘贵松) -《电子科技大学硕士论文》- 2012-03-01	0.5%（29） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第一章绪论

1.1课题背景和意义

1.1.1 课题背景

飞速发展的互联网技术推动人类社会的不断进步，但凡事皆有利弊，互联网技术的不断革新也促进了恶意代码的发展壮大，其危害也变得越来越难以控制。在人们所熟知的恶意代码中，尤以木马、病毒、蠕虫等恶意代码的传播最为广泛,社会各界深受其害，给个人、企业甚至政府造成了难以估量的损失。根据国内知名互联网安全厂商奇虎360发布的《2016年中国互联网安全报告》显示,仅2016年被确认检测到的就有1.9亿多的新增PC端恶意程序，可以预测实际新增的恶意程序将会远远大于这个数字，足以说明互联网安全问题已迫在眉睫。近年来，敲诈者病毒也频繁爆发，仅在中国国内就发生了两次规模较大的传播,遭受该病毒攻击的电脑用户至少达到了490多万，并且大部分受害者并不清楚何时感染的以及如何被感染的。相比于PC端，移动端的情况虽然好了很多但也不容乐观，仅2016年就截获了1400多万针对Android平台的新增恶意程序，大部分为资源消耗类，但是也出现了大量的勒索类软件，大约有17多万新增的勒索类软件，170多万台手机遭受攻击。相比于PC端，用户更习惯在移动端存储个人信息，不法分子也开始将目光聚焦在了截获盗取个人信息上，在截获的此类恶意代码中，窃取短信信息的占67.4%，窃取收集银行信息的占34.8%，窃取联系人信息的占了10.0%，窃取手机通话记录的占了3.7%，有2.0%的窃取了社交软件聊天记录。在信息共享技术高度发达的今天，人们对于个人隐私的保护也越来越关注。以下总结了2017年上半年发生的典型安全事件：

（1）2017年上半年爆发了多次攻击事件，比较知名的有“WannaCry”、“暗云III”、“Petya”等多种类型的病毒木马，为社会各界再次敲响了互联网安全的警钟。根据相关报告显示,仅在2017年上半年,就有超过10亿次的PC端病毒被腾讯安全反病毒实验室拦截，同2016年下半年相比增长了30%。其中，在“WannaCry”的刺激下，勒索类病毒增长明显，在一个季度之内就增长了13.39%，而最具传染力的勒索病毒“PolyRansom”占了所有勒索病毒的78.84%。

（2）2017年3月下旬，苹果公司遭受网络犯罪团伙的勒索，该犯罪团伙声称掌握了3亿多的用户账号信息，并且可以远程清除iCloud账户。

（3）2017年4月中旬，据外国的一家媒体hackread报道，1亿多条优酷账户信息的数据库在暗网被售卖。

（4）2017年4月下旬，有媒体报道12306官方网站再现安全漏洞，用户个人信息遭到泄露。

（5）2017年5月中旬，国务院某款App的页面被植入了色情广告，经过排查确定为当地运营商HTTP被劫持导致H5页面被

插入广告。

在众多的互联网安全事件中,尤以恶意代码最具威胁性,其带给社会各界的经济损失占很大比例，主要原因有三点。第一，技术较为成熟，恶意代码的出现可以追溯到上世纪八十年代初期，经过几十年的不断壮大与技术积累，恶意代码变得越来越复杂，其破坏力也不断增长；第二，犯罪成本低，由于信息共享技术的发展，恶意代码的开发工具也趋于自动化，并且极易获得，开发变得越来越方便，而传播方式也由原来的被动传播进化为主动传播，使得用户极易被感染，据统计80%以上的用户曾遭受过恶意代码的侵袭。第三，获利方便，恶意代码出现的初衷是为了技术炫耀，如今在利益的驱动下，更多的恶意代码作者有了经济利益诉求，促进了恶意代码的发展，这也是勒索软件出现的原因。现如今,提高互联网安全已经成为国家的一项重要基本战略。

1.1.2 课题意义

很大一部分网络安全事件发生都是由恶意代码引起的,并且根据以往案例来看,往往都是恶意代码造成一定的损失之后,针对该恶意代码的分析及检测技术才会被提出。出现这种情况的原因无非有两个:首先是恶意代码越来越复杂,并且种类繁多,传播形式多种多样,使得用户很容易被感染;其次是恶意代码检测技术不够成熟,尽管很多学者为恶意代码检测做了很多研究,也提出了很多检测方法,但是理论到应用总是需要时间,而这些时间给了恶意代码去进一步变异的可能。此消彼长下,恶意代码的危害始终存在,而检测与反检测的斗争不会停止。

近年来,恶意代码的使用与各种各样的经济甚至政治利益互相牵扯,其危害性和隐藏性日益增强,破坏的目标、目的以及要达成的后果更加具有针对性。现如今的恶意代码编写者已经不单单是为了技术炫耀，更多的恶意代码的编写者开始利用自己的技术，危害他人以达到经济或者政治诉求。据统计，中国木马产业链一年的收入已逾上百亿元,黑色产业链正在逐渐成型。研究出一种可以对恶意代码进行有效检测的方法已经迫在眉睫。

综上所述,恶意代码的危害无处不在,不仅给个人、企业带来了巨大的损失,甚至可能给国家安全带来不可预期的危害。对于个人,恶意代码的入侵会导致个人隐私暴露在开放的互联网环境下，这些信息可能被犯罪分子利用，造成个人的经济损失或名誉损失；对于企业，商业机密的泄露，如果是核心技术外泄，那么企业的竞争力将会大打折扣，对发展造成不良影响；对于国家而言，信息时代背景下，信息安全作为国家安全的重要组成部分，其扮演的角色也越来越重要，信息安全已经成为了国家的一项重要发展战略。因此研究更加有效的恶意代码检测技术是非常具有现实意义的。

1.2国内外研究现状

上世纪80年代，Apple II[2]作为真正意义上的第一个病毒诞生了，自此拉开了恶意代码检测与反检测的战争序幕，众多的学者开始投入大量的精力和时间到恶意代码的对抗中。矛与盾从来都是相互促进的，恶意代码的发展同样促进了检测技术的进步，目前已经有各种各样的检测技术被应用在了商业环境中。以下是国内外学者在恶意代码检测方面做的一些研究。

基于系统调用的检测方法。Sung[3]等人提出了一种针对恶意代码变种的静态检测技术，该方法的主要是基于系统调用来检测恶意代码的，首先将恶意代码通过逆向工程技术得到其汇编文本文件，然后通过分析汇编文本信息，得到系统调用序列，最后根据序列相似度判断该程序是否是恶意的。张波云[4]等人也提出了相似的检测方法，不同的是该方法在提取系统调用信息时程序处于运行状态，一般是通过在虚拟环境中执行恶意代码程序，提取系统的调用序列，然后使用n-gram算法提取调用序列特征，在检测恶意代码时程序处于运行状态的分析方法被称为动态分析方法。对于恶意代码的分析方法，恶意代码从来都不缺少应对手段，混淆技术的产生使得该方法的检测变得困难。

基于语义的检测方法。为了解决混淆技术带来的困惑，有学者提出了基于语义分析的检测方法，该方法试图分析指令运行时的语义，一般的分析方法有符号执行[5]、模型检验[6]、逻辑推理证明等。Cousot.P 和 Cousot.R[7]等人提出了程序分析构造和逼近不动点语义理论，使得程序的语义分析有了理论依据。M.Christodorescu[8]使用行为自动机表述恶意代码，并引入了一种抽象模式库来表示自动机的符号，最终恶意代码将会被表达为库中符号表示的自动机，最后使用模型检验实现预测。之后他又提出了一种基于语义的恶意代码检测方法，恶意代码的行为通过迹语义来描述，采用抽象解释方法检测恶意代码的行为[9]。D.Preda[10]同样使用了抽象解释的思想对恶意代码进行检测，再度证明了该方法对于混淆技术具有一定的对抗能力。Singh[11]通过分析存在于汇编文本中的数据流信息，利用线性时态逻辑语义模型检测程序是否存在恶意行为。Kinder[12]结合程序流程图和函数之间的调用关系，使用计算逻辑树描述恶意代码。李佳静等人[13]通过分析函数调用及其调用序列之间的依赖关系，使用有穷自动机描述程序的行为信息。王晓洁和王海峰[14]通过语义描述恶意代码的行为，实验证明该方法对使用了混淆技术的恶意代码也具有相当明显的检测效果。

基于签名的检测方法。孔德光等人[15]提出了一种签名提取算法，对于恶意代码检测的准确率有了较大的提高。G.Tahan等人[16]提出了一种自动签名提取算法，该算法主要被应用在高速恶意代码的过滤装置中。Y.Tang等人[17]提出了一种基于正则表达式的签名算法，这种方法能否更加准确的产生基于漏洞的签名。Y.Chen[18]等人提出了在网络层没有任何主机分析的蠕虫执行的脆弱性驱动的签名，实验效果非常好。目前所有的基于签名的恶意代码检测技术，通常准确率都非常高，但是其对于未知的恶意代码无能为力，并且需要不断的更新病毒库，随着系统的使用，其性能问题也会变得越来越明显。

基于操作码的检测方法。为了提高分类的准确性，越来越多的学者使用n-gram序列的字节序列或者操作码序列代替二进制序列作为特征。Robert等人[19]使用操作码序列作为代表恶意代码的特征，然后使用文本分类算法对恶意代码进行分类。Schultz等人[21]第一次提出使用机器学习算法对恶意代码进行检测，研究表明机器学习算法对于恶意代码的分类具有显著的效果。Kolter[22]同样使用基于机器学习算法的检测方法，不同的是他使用了n-gram 算法的字节序列作为恶意代码的特征表示。文献[23]的作者同样在特征的提取方式上做了创新，首先使用n-gram算法提取特征，然后使用信息增益的方法选择对分类有帮

助的特征，并使用了多种机器学习算法对恶意代码进行分类，并且取得了不错的实验效果。Kolterh和Maloof[22]对恶意代码的家族分类做了研究，根据恶意代码的功能行为对恶意代码进行更加细致的划分，可以帮助学者更加方便的分析每一种恶意代码，发现其性从而帮助分类。文献[24]中作者针对恶意代码的变种，提出了一种层次特征选择方法，使用n-gram算法提取特征后，选择那些出现频率较高的特征作为最终的特征。Raja等人[25]同样是在特征的选择方面做了创新，提取到汇编操作码系列之后，再使用CPD（Categorical Proportional Difference）方法选择分类较好的特征。Dolev[26]使用操作码作为恶意代码的特征表示。近年来，操作码特征已经被用来检测蠕虫的变种和一些间谍软件[27]。也有学者将操作码和签名技术结合来检测恶意代码的变种。文献[19]首先提取操作码，然后根据其序列检测未知恶意代码。文献[28]中的作者首先使用了变长的指令序列作为恶意代码的特征，并使用bagging算法得到了很好的效果。也有学者选择使用十六进制码作为特征进行恶意代码的检测[29]。

基于控制流的检测算法。文献[30]中作者使用了程序控制流设计了一个分类算法，也取得了不错的分类效果。Ismail B[31]将控制流程图和函数调用拓扑图结合起来实现恶意代码的分类。Halvar[32]利用了程序控制流程图的拓扑图的同构来实现检测。这种算法能够很好的检测恶意代码的变种，因为同一种类的恶意代码其拓扑图基本相似。

基于数学统计的检测算法。Igor[33]统计了操作码序列出现的频率，然后挖掘出序列之间的相关性，并依此对恶意代码进行分类，分类效果相当明显。Perdisci等人[34]利用PE文件提取特征并使用不同的机器学习算法进行分类，由于效果比较明显，他们根据这种方法开发出了一款恶意代码的检测工具[35]。

现有的恶意代码检测技术有很多，各有优劣。在充分借鉴已有研究的基础上，本文对基于机器学习算法的恶意代码检测方法做了改进，并且得到了很好的检测效果。

1.3研究内容

本文改进了一种基于机器学习算法的恶意代码检测方法，并基于该方法实现了一个恶意代码检测系统。方法主要是使用汇编操作码的抽象化技术将汇编码表示为中间码，并结合Eclat算法对中间码的频繁项集进行分析，预测出一种最优的中间码序列，然后使用n-gram算法提取中间码特征序列生成概率矩阵，作为代表恶意代码的特征，最后使用机器学习算法进行建模。当抽象方式有多种的时候，这种方法能够针对n-gram算法提取出的中间码特征序列，预测出一种对分类效果最好中间码特征序列作为生成概率矩阵的依据。

在仿真实验中，本文提供了两种抽象方式，分别记为Abs1和Abs2，Abs1是根据作者的理解对汇编码提出的抽象方式，Abs2是文献[1]中提出了抽象方式，本文将会分别使用本文方法和文献方法进行实验并分析对比实验结果，给出结论。实验主要步骤有：首先，为了逃避病毒检测系统的检测，一般的恶意代码作者都会对恶意程序进行加壳处理，所以本文的第一步就是对恶意代码进行查壳和脱壳处理；其次对恶意代码样本进行反汇编处理，得到样本的汇编文本，并从中提取出汇编操作码序列；然后根据已配置的抽象方式对汇编操作码序列进行抽象化处理，得到各抽象方式对应的中间码序列；接着使用n-gram算法获得中间码特征序列，并对中间码特征序列的频繁项集进行分析，预测对分类效果最明显的中间码特征序列，并依据选择出的中间码特征序列生成概率矩阵；最后对比了随机森林算法、支持向量机以及K邻近三种机器学习算法使用概率矩阵作为输入的分类效果，结果显示随机森林算法效果最为显著和稳定。另外根据本文改进的方法设计并实现了一个恶意代码检测系统。本文的工作有以下几点：

第一：收集实验样本，并对样本进行预处理操作，然后提取基于文本的汇编操作码序列。

第二：本文改进了一种基于机器学习算法的恶意代码检测方法。通过汇编操作码抽象化技术、Eclat算法频繁项集分析和n-gram算法提取特征序列获得概率矩阵，并以此作为代表恶意代码的特征，最后使用机器学习算法构建分类模型。

第三：对本文改进的方法进行实验仿真，并对实验结果进行对比分析，得出结论。

第四：针对本文提出的改进方法，结合实验仿真的结果分析，采用模块化编程技术实现了一个恶意代码检测系统。

1.4论文结构

本文总共分为四章：

第一章为绪论，主要介绍课题背景及意义。并且详细阐述了国内外对于恶意代码检测技术的研究现状和存在的问题，最后介绍了本文的研究内容和章节安排。

第二章是相关理论与关键技术。首先介绍了恶意代码的定义和分类，然后对现有恶意代码分析技术、检测技术和反检测技术做了相关介绍。

第三章是基于机器学习算法的恶意代码检测方法。首先对汇编操作码的特点进行了分析，并介绍了有关学者对汇编码抽象化的一些研究；然后概要的介绍了本文方法的大体流程，接着对方法的各部分进行了详细说明，包括数据预处理、概率矩阵的生成过程和恶意代码的分类；最后根据本文方法进行了实验仿真，并和传统方法的实验结果进行了对比分析，得出结论。

第四章是系统设计与实现。首先介绍整个系统的架构设计，包括系统的功能分析及组成模块，并对各模块的功能进行了简单介绍；接着对各模块的实现方式进行了详细介绍；最后对本章进行总结。

第五章为总结与展望。概括总结了本文的主要研究成果和不足，对未来的可研究方向进行了展望。

指　标
疑似剽窃文字表述

1. 背景和意义

1.1.1 课题背景

飞速发展的互联网技术推动人类社会的不断进步，

2. 本文的第一步就是对恶意代码进行查壳和脱壳处理；其次对恶意代码样本进行反汇编处理，得到样本的汇编文本，并从中提取出汇编操作码序列；

3. 论文结构

本文总共分为四章：

第一章为绪论，主要介绍课题背景及意义。并且详细阐述了国内外对于恶意代码检测技术的研究现状和存在的问题，最后介绍了本文的研究内容和章节安排。

第二章是

3. 第二章相关理论与关键技术

总字数：6497

相似文献列表　文字复制比：44%(2857)　疑似剽窃观点：(0)

1	基于操作码序列的静态恶意代码检测方法的研究 卢占军(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2012-12-01	17.9%（1164） 是否引证：否
2	10S051019-卢占军-丁宇新 卢占军 -《学术论文联合比对库》- 2012-12-12	17.8%（1154） 是否引证：否
3	BH2009925453 -《学术论文联合比对库》- 2012-12-11	17.8%（1154） 是否引证：否
4	BH2009921397 -《学术论文联合比对库》- 2012-12-04	16.6%（1077） 是否引证：否
5	094616006_冯本慧 冯本慧 -《学术论文联合比对库》- 2013-09-23	14.1%（916） 是否引证：否
6	基于数据挖掘与机器学习的恶意代码检测技术研究 冯本慧(导师：王加阳) -《中南大学硕士论文》- 2013-09-01	12.6%（817） 是否引证：否
7	结合语义的统计机器学习方法在代码安全中应用研究 孔德光(导师：奚宏生) -《中国科学技术大学博士论文》- 2010-05-01	9.5%（614） 是否引证：是
8	基于操作码行为深度学习的恶意代码检测方法 陈晨(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2013-12-01	6.8%（444） 是否引证：否
9	11S051036-陈晨-丁宇新 陈晨 -《学术论文联合比对库》- 2013-11-27	5.8%（378） 是否引证：否
10	11S051036-陈晨-丁宇新 陈晨 -《学术论文联合比对库》- 2013-11-14	5.8%（378） 是否引证：否
11	恶意代码行为挖掘关键技术研究 解培岱(导师：卢锡城) -《国防科学技术大学博士论文》- 2013-10-01	3.8%（249） 是否引证：否
12	基于沙箱技术的恶意代码行为自动化检测方法 李志勇(导师：李伟明) -《华中科技大学硕士论文》- 2015-05-01	2.2%（140） 是否引证：否
13	某某_基于某某某技术 某某 -《学术论文联合比对库》- 2013-10-11	1.4%（92） 是否引证：否
14	基于污点跟踪的固件漏洞定位研究 戴忠华;费永康;赵波;王婷; -《山东大学学报(理学版)》- 2016-04-18 1	0.5%（30） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第二章相关理论与关键技术

本章首先介绍了恶意代码相关概念，并对现有的恶意代码检测技术以及反检测技术进行了详细介绍；然后对恶意代码的反检测技术做了相关介绍；最后对恶意代码的分析技术做了总结。

2.1 恶意代码简介

2.1.1 恶意代码的定义

恶意代码也成为恶意软件，是对各种敌对和入侵软件的概括性术语。包括各种形式的计算机病毒、蠕虫、特洛伊木马、勒索软件、间谍软件、广告软件以及其他的恶意软件。形式上多种多样，可以是可执行文件、脚本、插件等等。其违背使用者的意愿去执行一些操作，损害用户的利益以达到入侵者不可告人的目的。

2.1.2 恶意代码的分类

根据不同的依据，恶意代码有很多种不同的分类方法，没有一种标准的分法，但是常见的种类有：计算机病毒、蠕虫、特洛伊木马、间谍软件、勒索软件等等。下面对几种恶意代码做简要介绍：

(1)计算机病毒。第一份关于计算机病毒理论的学术报告由冯·诺伊曼完成，他描述病毒是一个可以复制其自身的程序，但“病

毒”一词最早出现在Fred Cohen于1984年的论文《电脑病毒实验》中。为了能够对自身进行复制，病毒必须掌握对内存的写操作，所以大部分病毒都是依附在合法的可执行文件中，当宿主文件运行时，病毒才有机会运行自身。一般而言病毒根据运行时的状态可分为常驻行病毒和非常驻行病毒。非常驻行病毒包括两个模块：搜索模块和复制模块，搜索模块会尽可能的搜索其他宿主文件，并调用复制模块对该宿主进行感染；常驻行病毒只包含复制模块，该模块并不会被搜索模块调用，而是当操作系统运行特定动作时触发复制模块，从而对文件进行感染。由于Windows系统时当今世界使用最多的操作系统，所以大部分病毒都是针对Windows系统的，而且早期的Windows系统并没有有效的安全与防护功能，这也为Windows病毒的泛滥提供了温床。病毒一般具有传播性、隐蔽性、感染性和潜伏性等特征。

（2）特洛伊木马。计算机木马和病毒一样都是一种有害的程序，其具有一定的伪装型，表面上没有危害，却在用户不经意间对于系统实行破坏或折窃取数据。一个完整的特洛伊木马包括两个部分：服务端和客户端。将服务端植入到受害者的电脑，并伪装成一种正常进程运行在服务端，暗自打开一些端口，客户端利用这些端口连接到服务端进行入侵。特洛伊木马不经用户同意就可以获取电脑的控制权，因为运行在服务端的程序功能比较简单，所以容量十分轻小，运行时并不会占用系统过多资源，因此也难以被察觉。木马的技术发展迅速，至今已经经历了六代的改进，PassCopy和暗黑蜘蛛侠是这类木马的代表。

（3）蠕虫。蠕虫病毒是一种常见的计算机病毒，具有自我复制和传播的特性，与一般病毒不同的是，蠕虫并不需要依附在宿主文件中。有两种不同类型的蠕虫：主机蠕虫和网络蠕虫，主机蠕虫完全包含在其运行的计算机中，当通过网络传播到其他主机后，就会终止它自身，即任何时刻蠕虫只有一个实例在运行，而网络蠕虫就没有这种限制，它可以在同一时刻有任意多个实例在运行。著名的“熊猫烧香”就是蠕虫病毒的一种。

（4）后门。后门程序是留在计算机系统中，供某些人通过特殊方式控制计算机系统的途径。它跟木马的共同点是都是隐藏在用户系统中，暗地里向外发送信息，并且本身有一定的权限；不同点在于木马是一个完整的程序，而后门一般来说功能简单且单一。后门一般不具有自我复制的功能，也就是后门程序并不会感染其他主机。

（5）Rootkit。Rootkit一词最早出现在Unix系统上，最早的Rootkit是用于善意用途的，但后来Rootkit被黑客用于入侵和攻击他人电脑，病毒、软件等也常用Rootkit来隐藏踪迹。现在Rootkit被视为一项技术，更多的是被作为驱动程序，加载到操作系统内核中的恶意软件。因为其运行在特权模式下，它可以帮助攻击者获取主机管理权限，实现维持拥有管理权限的程序[36]。

（6）间谍软件。它是指在用户不知情的情况下搜集用户个人信息的计算机程序。间谍软件一般采用多种技术记录个人信息，例如键盘录制、记录用户访问Internet行为等。间谍软件的用途也多种多样，比如盗取用户的银行卡信息和密码或者统计用户网络行为作为广告用途等。

2.2 恶意代码检测与反检测技术

2.2.1 恶意代码检测技术

目前最为成熟的恶意代码检测技术是基于“特征码”的，基本的检测思路是，当截获新的恶意代码时，对其进行全面分析，提取可以代表它的特征码，然后将该特征码记录在数据库中，如果有程序的特征码在数据库中被检测到，则判定其为恶意代码。目前应用于商业的恶意代码检测软件大部分使用的就是该技术。但是这种技术最大的缺点就是无法被未知或变种的恶意代码进行有效的检测。因此有学者提出了基于启发式的检测算法。根据对恶意代码分析原理的不同将恶意代码检测技术分为以下几类：

（1）基于特征码的检测技术

基于特征码的检测技术是一种使用时间最长并且应用范围最广的一种技术。被国内外许多知名的恶意代码检测厂商所使用，因其使用简单、开销小等特点，被广泛应用于文件类型的病毒检测中。检测软件的核心就是恶意代码特征库的完全性，当需要检测某个程序是为恶意代码时，就对该程序启动特征扫描，然后提取特征，再与特征库进行匹配，如果匹配成功，则表明该程序是一种恶意代码。该技术的关键在于如何选取最能代表恶意程序的特征值。采用该方法，检验结果准确，鲜有误报情况，但该方法对于未知或者变形恶意代码无能为力。并且这种方法随着使用时间的增长，特征库的规模不断增加，需要用户经常对特征库进行更新，大大影响检测的速度和系统的性能。

（2）基于行为的检测技术

基于行为的检测技术是利用恶意代码的特有行为对恶意代码进行检测。恶意代码在许多行为模式上区别于良性程序，比如特定的系统调用或者系统调用序列，因为恶意代码要对系统实施破坏，就必须获得系统的高级权限，这样才有可能对系统资源进行调用[37]。通过分析程序的行为模式，从而完成对程序的检测。当程序在运行状态时，系统对其行为进行监控，若发现壳可疑行为，就立即向操作系统或者用户发出告警。一般用于检测恶意代码的行为特征如下：

1) 对特定文件执行写操作：有些恶意代码时依附而生，所以在其执行时，就要将自身代码附加在感染文件中，可以监控是否有异常写操作。

2) 监控系统调用序列：某些系统调用序列可以体现某种程度的程序语义。系统调用是用户态和内核态的唯一接口，恶意代码想要获取高级权限实施破坏行为，就必然要经过系统调用接口。

3) 修改内存总量：恶意代码为了完成特定的恶意意图，经常会常驻在内存中，并且不能被覆盖，那么将会减少系统内存的总量，使得该段内存不受系统内核控制。

（3）基于启发式的检测技术

启发式检测技术预先为恶意代码特征设置了一个阈值。扫描完文件后，当提取的特征与恶意代码特征的相似度达到一定值

时，判定该文件为恶意代码。例如，一些恶意代码总是会调用一些内核函数。通常，这些调用的顺序是一定的规则。因此，通过分析内核函数名称和调用次数，恶意代码可以构造一个内核函数的特征。启发式方法是主动防御技术，对未知的恶意代码检测有明显的效果。因此，这种方法在商业开发中被重点应用。启发式测试可以分为静态和动态启发。

实际上，静态启发式方法是传统特征识别方法的延伸。通过分析程序的系统API调用顺序为特征，也有领域的一些专家根据自己的经验研究和总结了一些恶意代码的行为特征。监控时，一旦发现此类行为特征，将立即报警并进行相应的处理。该方法能够有效地检测已知的恶意代码，并检测出一些未知的恶意代码，但是在发现恶意代码时，系统经常被感染。另外，行为检测是对系统的实时监控，可能会继续占用大量的内存，CPU等系统资源。在商业领域，这种方法主要用于辅助检测。

动态启发式技术的主要工作原理是将一个单独的虚拟环境划分为一个计算机系统。当发现一个可疑的程序时，它不会立即停止，而是让它继续运行。“沙盒”技术是一种动态的启发式方法，“沙盒”记录可疑程序的行为，直到恶意代码完全暴露，就执行回滚以使计算机恢复到执行可疑程序之前的状态。近年来，病毒检测公司纷纷将“沙盒”技术应用于商业查杀工具中进行推广。

（4）基于语义的检测技术

基于语义的检测技术是目前恶意代码检测的研究热点。混淆技术的出现，为传统的恶意代码检测技术提供了阻碍，虽然该技术在程序中插入了大量的垃圾指令或者随意改变指令的顺序，但是其要完成的逻辑功能并未发生改变。通过对恶意代码进行分析，抽象出程序指令行为并建立行为模型，不仅可疑描述恶意程序的基本行为，而且具有较强的泛化能力。正是因为它具有强大的泛化能力，使得在检测恶意程序的变体是变得更加容易。另外，它还可以被用来检测未知类型的恶意代码。现阶段基于语义的检测方法分为基于内存和函数调用的方法。 M.Christodorescu[8][9]提出了一套抽象理论和语义框架，用自动机来描述程序的行为，通过抽象理论抽象出程序的行为并建立抽象模型库，并使用该抽象库作为自动机的符号表，最后将恶意行为描述为自动机表示的模板，最后使用模型检测的方法来检测样本是否包含恶意行为。模型检测遍历系统的所有状态空间，以查看是否存在匹配的路径状态。后来他又提出了迹语义学的概念，迹语义学作为程序的基本语义，并定义了等价条件，给出了近似检测算法的抽象解释。抽象解释理论为解决不确定和复杂问题的近似提供了一种很好的构建方法。基于函数调用的方法是提取程序中使用的函数，并结合程序的控制流程图，通过同构，线性时态逻辑，计算逻辑树，有限状态机和下推自动机等方法进行描述恶意行为，最终通过模型测试来完成对恶意代码的检测。

（5）基于机器学习的检测技术

基于机器学习和数据挖掘的检测方法。随着检测技术的不断发展，机器学习和数据挖掘的方法已经被应用在恶意代码检测领域。主要应用分类、关联规则挖掘、序列模式分析以及聚类等。主要思想是利用数据挖掘技术从现有的数据中挖掘一些有意义的模式，利用机器学习算法总结现有样本特征，然后根据特征的相似性等完成分类的任务。其中最主要的是选择好的特征和有效的分类器。检测步骤如下：首先，分析样本以确定提取哪些特征或者特征序列；其次，根据提取的特征的特点，选择合适的特征选择方法从所有提取的特征中选择一些分类效果好的特征；最后，根据实际情况选择较好的模型实现分类。

2.2.2 恶意代码反检测技术

恶意代码检测与反检测技术总是相互促进，相辅相成。检测技术的进步也带动了反检测技术的发展，当前出现了各种各样的恶意代码反检测技术，现总结如下：

（1）加壳技术

恶意代码作者为了防止自己的程序被检测软件发现，利用一些软件技术给恶意代码加外壳，可以是利用算法将自己伪装成正常程序，或者利用特殊的算法将自己压缩或加密，使得检测软件很难检测。但是这些“壳”都有一个特点就是，他们先于程序获得执行控制权，然后把伪装后的程序还原，再把执行权交还给原始代码：是一类自修改代码。

（2）反虚拟执行技术

不可否认，虚拟执行的系统和真实系统或多或少存在差异[41]。例如，在硬件上，调试器总是设置硬件断点，而虚拟机总是在模拟硬件，这与真实的硬件是不同的; 虚拟机和实际机器的执行环境和内核地址空间是不同的，调试器也必须插桩在某个进程上才能起到监控的作用; 应用程序，虚拟机和调试器都有外部应用程序，对进程可见，用于检查运行环境。还有一些指令，其在虚拟环境中的执行时间总是比真实环境长，频繁执行这些指令的程序可以指示它运行一个虚拟机环境。

（3）代码迷惑技术

恶意代码迷惑技术是指通过某种程序代码变换，改变自身在空间和时间上的结构，但是完成相同的逻辑功能。恶意代码在进行迷惑处理之后，使得逆向工程分析变得难以进行。迷惑技术本身是一种保护软件的手段，但是常常被用来对抗分析和检测。恶意代码的迷惑技术可以有效的对抗恶意代码的静态分析技术和动态反汇编技术。目前主要有基于加密的迷惑技术和基于代码变换的迷惑术。其中代码变换主要指在源程序中，利用等价指令替换、指令位置交换、添加新指令等手段改变程序形式，但逻辑功能保持不变。

2.3 恶意代码分析技术

恶意代码分析是确定恶意代码意图的过程，是实行恶意代码检测的必要前提。恶意代码分析的直接结果是用于实现恶意行为建模的元数据信息，如指令流、API调用序列等，为后续恶意代码的检测工作提供必要的支持。

恶意代码的分析技术一般可分为静态分析和动态分析。

（1）静态分析技术

静态分析技术是指对被测软件的源程序或者二进制码进行扫描，从语法、语义的层面去理解程序的行为，以期望获取程序

在运行过程中的信息，而不需要运行程序。

要进行恶意代码的静态分析，首先需要对恶意程序进行反汇编，常用的反汇编工具有：W32DASM、objdump、PEid、HIEW、IDA Pro等。

静态分析技术由于不会运行程序，因此不会对计算机系统造成任何伤害，其分析效率相对动态分析而言较高，同时由于静态分析技术从程序本身入手，因此可以获得程序的全部信息，分析结果较为全面。但是由于静态分析技术的前提条件是对程序进行正确的反汇编，现如今很多恶意代码编写者常常会对恶意代码进行加壳、加密或者压缩使得恶意程序很难被正确的反汇编。总之，如果恶意代码无法被正确的反汇编，那么静态分析将会失效。

（2）动态分析技术

动态分析技术是指在受控环境下对程序的实际运行情况进行监控，监控程序行为的执行情况，记录程序信息的执行情况。由于动态分析需要先运行程序，为了防止恶意代码破坏当前环境，系统在执行恶意代码之后，会自动恢复到未执行恶意代码前的原始状态，以防止恶意代码的相关信息影响下一步的分析结果。因为动态分析技术则可以得到真实的恶意代码执行信息，所以可以有效地解决加壳和加密的干扰。

动态污点分析技术[38]是当前最热门的动态分析技术，其基本原理是将所有不可信的外部数据标记为污迹，然后追踪标记为污迹的数据的扩散，并记录系统调用或指令执行等相关信息，然后利用这些信息进行检测。动态污点分析可以记录恶意代码的细粒度特征，是一种非常流行的动态恶意代码检测技术。

动态分析技术也有一些缺点，比如开销大，一次只有一个路径，处理恶意代码中的多路径问题的代码不正确，以及无法模拟完全真实的计算机环境并且对某些环境敏感的恶意代码没有进行有效的检测，因为这些恶意代码可以检测到虚拟机或模拟器的存在，以隐藏自己的真实行为，并且不知道某些恶意代码何时会触发。 动态分析技术也受到行为层的影响。混淆技术会干扰[39]，如等效行为替换，模拟序列或混淆序列。所以有学者开始尝试将静态分析和动态分析相结合[40]进行恶意代码检测，充分利用这两种分析技术的优势。

两种分析技术各有利弊，静态分析技术开销较小，关注的是恶意程序本身的语法或结构特征；动态分析技术开销大，关注的是恶意代码的行为特征。静态分析在分析上是全面的，并提供了有关恶意代码的完整信息，但获取功能的方式通常是非导向的，因此可能包含大量无用的信息，并且容易受到代码混淆技术的影响。动态分析切合实际行为信息，但是一次只能获得一个与当前测试环境相关的行为，信息不够全面。目前应用最广泛的技术还是静态分析技术。简而言之，无论是静态分析还是动态分析，都需要利用恶意代码分析技术和监控技术来获取恶意代码的基本属性和执行信息，进一步了解恶意代码的功能，实现恶意代码的检测并抑制恶意代码。

2.4 本章小结

本章首先介绍了恶意代码的定义以及恶意代码的分类，具体介绍了病毒、特洛伊木马、蠕虫等恶意代码的特征和危害。其次，介绍了恶意代码的检测技术，详细阐述了基于特征码的检测技术、基于行为的检测技术、基于启发式的检测技术、基于语义的检测技术和基于机器学习的检测技术。最后，针对目前主流的恶意代码反检测手段以及分析技术做了详细说明。

指　标
疑似剽窃观点
1. 需要利用恶意代码分析技术和监控技术来获取恶意代码的基本属性和执行信息，进一步了解恶意代码的功能，实现恶意代码的检测并抑制恶意代码。
疑似剽窃文字表述
1. 技术做了总结。 2.1 恶意代码简介 2.1.1 恶意代码的定义 恶意代码 2. 包括各种形式的计算机病毒、蠕虫、特洛伊木马、勒索软件、间谍软件、广告软件以及其他的恶意软件。 3. 2.1.2 恶意代码的分类 根据不同的依据，恶意代码有很多种不同的分类方法，没有一种标准的分法，但是常见的种类有：计算机病毒、蠕虫、特洛伊木马、间谍软件、勒索软件等等。下面对几种恶意代码做简要介绍： (1)计算机病毒。 4. 技术最大的缺点就是无法被未知或变种的恶意代码进行有效的检测。因此有学者提出了基于启发式的检测算法。 5. 方法对于未知或者变形恶意代码无能为力。并且这种方法随着使用时间的增长，特征库的规模不断增加，需要用户经常 6. 判定该文件为恶意代码。例如，一些恶意代码总是会调用一些内核函数。通常，这些调用的顺序 7. 分析，抽象出程序指令行为并建立行为模型，不仅可疑描述恶意程序的基本行为，而且具有较强的泛化能力。正是因为它具有强大的泛化能力，使得在检测恶意程序的变体是变得更加容易。另外，它还可以被用来检测未知类型的恶意代码。现阶段基于语义的检测方法分为基于内存和函数调用的方法。

8. 作为程序的基本语义，并定义了等价条件，给出了近似检测算法的抽象解释。抽象解释理论为解决不确定和复杂问题的近似提供了一种很好的构建方法。基于函数调用的方法是提取程序中使用的函数，并结合程序的控制流程图，通过同构，线性时态逻辑，计算逻辑树，有限状态机和下推自动机等方法进行描述恶意行为，最终通过模型测试来完成对恶意代码的检测。
9. 随着检测技术的不断发展，机器学习和数据挖掘的方法已经被应用在恶意代码检测领域。主要应用分类、关联规则挖掘、序列模式分析以及聚类等。主要思想是利用数据挖掘技术从现有的数据中挖掘一些有意义的模式，利用机器学习算法总结现有样本特征，然后根据特征的相似性等完成分类的任务。其中最主要的是选择好的特征和有效的分类器。检测步骤如下：首先，分析样本以确定提取哪些特征或者特征序列；其次，根据提取的特征的特点，选择合适的特征选择方法从所有提取的特征中选择一些分类效果好的特征；最后，根据实际情况选择较好的模型实现分类。
10. 结果是用于实现恶意行为建模的元数据信息，如指令流、API调用序列等，为后续恶意代码的检测工作提供必要的支持。
恶意代码的分析技术一般可分为静态分析和动态分析。
11. 工具有：W32DASM、objdump、PEid、HIEW、IDA Pro等。
静态分析技术由于不会运行程序，因此不会对计算机系统造成任何伤害，其分析效率相对动态分析而言较高，同时由于静态分析技术从程序本身入手，因此可以获得程序的全部信息，分析结果较为全面。但是由于静态分析技术的
12. 由于动态分析需要先运行程序，为了防止恶意代码破坏当前环境，系统在执行恶意代码之后，会自动恢复到未执行
13. 动态污点分析可以记录恶意代码的细粒度特征，是一种非常流行的动态恶意代码检测技术。
动态分析技术也有一些缺点，比如开销大，一次只有一个路径，处理恶意代码中的多路径问题的代码不正确，以及无法模拟完全真实的计算机环境并且对某些环境敏感的恶意代码
14. 存在，以隐藏自己的真实行为，并且不知道某些恶意代码何时会触发。 动态分析技术也受到行为层的影响。
15. 开销较小，关注的是恶意程序本身的语法或结构特征；动态分析技术开销大，关注的是恶意代码的行为特征。静态分析在分析上是全面的，并提供了有关恶意代码的完整信息，但获取功能的方式通常是非导向的，因此可能包含大量无用的信息，并且容易受到代码混淆技术的影响。动态分析
16. 2.4 本章小结
本章首先介绍了恶意代码的定义以及恶意代码的分类，具体介绍了病毒、特洛伊木马、蠕虫
17. 检测技术，详细阐述了基于特征码的检测技术、基于行为的检测技术、基于启发式的检测技术、基于语义的检测技术和基于机器学习的检测技术。最后，针对目前主流的恶意代码反检测

4. 第三章基于机器学习算法的恶意代码检测方法_第1部分

总字数：9790

相似文献列表　文字复制比：10.7%(1052)　疑似剽窃观点：(0)

1	10S051019-卢占军-丁宇新 卢占军 - 《学术论文联合比对库》- 2012-12-12	8.4%（825） 是否引证：否
2	BH2009925453 -《学术论文联合比对库》- 2012-12-11	8.4%（825） 是否引证：否
3	基于操作码序列的静态恶意代码检测方法的研究 卢占军(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2012-12-01	8.3%（814） 是否引证：否
4	BH2009921397 -《学术论文联合比对库》- 2012-12-04	7.9%（774） 是否引证：否
5	基于操作码行为深度学习的恶意代码检测方法 陈晨(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2013-12-01	4.7%（462） 是否引证：否
6	11S051036-陈晨-丁宇新 陈晨 - 《学术论文联合比对库》- 2013-11-27	4.5%（442） 是否引证：否
7	11S051036-陈晨-丁宇新 陈晨 - 《学术论文联合比对库》- 2013-11-14	4.5%（442） 是否引证：否
8	入侵检测中神经网络融合学习方法的研究 吴静(导师：刘衍珩) -《吉林大学博士论文》- 2010-06-01	0.9%（86） 是否引证：否
9	基于深度表征的网络异常检测模型研究 薛成龙(导师：丁宇新) -《哈尔滨工业大学硕士论文》- 2014-12-01	0.7%（64） 是否引证：否
10	12s051010-薛成龙-丁宇新 薛成龙 - 《学术论文联合比对库》- 2014-12-05	0.7%（64） 是否引证：否
11	基于深度表征的入侵异常检测研究_薛成龙 薛成龙 - 《学术论文联合比对库》- 2014-11-06	0.7%（64） 是否引证：否
12	基于深度表征的网络异常检测模型研究_基于深度表征的入侵异常检测研究 薛成龙 - 《学术论文联合比对库》- 2014-11-05	0.7%（64） 是否引证：否
13	浅谈病毒与反病毒	0.6%（55）

第三章基于机器学习算法的恶意代码检测方法

本章对汇编操作码的特点进行分析，研究了众多的汇编码抽象方式后，改进了一种基于机器学习算法的恶意代码检测方法。该方法主要包括三个部分：数据预处理、概率矩阵的生成以及恶意代码分类，然后分别对方法的三个部分进行了详细介绍，其中概率矩阵的生成方法是本章的核心，也是本方法与其他传统方法的主要区别。最后对本文方法进行了实验仿真，证明本方法对于恶意代码的检测具有实际意义。

3.1 汇编操作码特点分析

大部分的基于机器学习算法的恶意代码静态检测技术都会使用汇编码进行分类模型构建，汇编操作码有以下特点：

第一，汇编语言是面向机器的程序语言，是一种用文字助记符来表示机器指令的符号语言，它是目前所有编程语言中最接近机器码的一种语言。它的针对性特别强，需要对机器硬件进行精确的控制，所以它的每一条指令都是极致细化的，这也导致了汇编操作码的局限性。当使用n-gram算法时，一般来说n的取值不会太大，n条少量的汇编操作码并不能反映出程序行为的一般特征。如果对汇编操作码进行抽象，得到更高维度抽象的中间码，同样是n条中间码，其反映的行为特征将会更加明确及有意义，所以对汇编码进行抽象对于恶意代码的检测是存在积极意义的。

第二，将汇编操作码进行抽象化处理能够显著的提升实验的效率。本文使用概率矩阵作为恶意代码的特征，即机器学习的输入，而x86汇编操作码的种类就有100多种，当使用2-gram、3-gram、4-gram算法时，概率矩阵的列数将分别是1002、1003、1004，即使后续对概率矩阵进行降维操作也是非常低效的。

针对以上对汇编操作码特点的分析，Kranz J等人[42]提出了一种名为GDSL的中间码来对x86汇编指令做出抽象，以期望表达更多的程序语义信息，Alam S等人[43]提出了一种被称为MAIL（Malware Analysis Intermediate Language，MAIL）的中间码表示方法，这些方法的确能够很好的归纳汇编指令的特征，但是王冰和方勇[1]认为其精简度不够，所以提出了一种新的指令抽象方式并且应用到了恶意代码检测领域，取得了不错的检测效果。

可以看出对于汇编指令的抽象化方式仁者见仁，很难去判定某种抽象方式绝对优于另一种抽象方式，不过可以肯定的是，任何一种指令抽象方式都有其适用范围，那么如何在复杂的互联网环境中保证检测系统始终选择最适合抽象化方式是本章要解决的问题。

3.2 基于机器学习算法的恶意代码检测方法流程

基于机器学习算法的恶意代码检测方法一般流程如图3-1。首先对样本进行预处理，该阶段包括对样本进行查壳脱壳以及对样本反汇编得到其对应的汇编文本；然后对汇编文本直接使用n-gram算法提取特征作为机器学习算法构造预测模型的输入；最后使用预测模型对恶意代码进行预测，得到预测结果。

图3-1 一般算法流程

由上一节分析可知，汇编操作码直接应用于恶意代码的检测时会有一定的局限性，所以本文改进了一种恶意代码的检测方法，主要流程如图3-2。

图3-2 本文算法主要流程

首先是样本预处理，该步骤包括对所有的样本进行查壳脱壳处理，并对脱壳后的样本进行反汇编，得到其汇编文本文件；其次，根据汇编文本文件提取汇编操作码，并根据不同的抽象方式对汇编操作码进行抽象，得到对应的中间码序列；然后使用n-gram算法提取样本的中间码特征，接着对各中间码特征序列进行频繁项集的相似度分析，选择相似度最小的中间码特征序列，作为生成概率矩阵的依据；最后，使用机器学习算法完成分类检测。

3.3 数据预处理

数据预处理阶段和一般检测算法并没有多大区别，包括恶意代码的查壳与脱壳、反汇编。

3.3.1 查壳与脱壳

壳是指在计算机软件中一段专门保护软件不被非法修改或反编译的程序。它们一般都先于程序运行，拿到控制权，然后完成保护软件的功能。

壳通常分为两类：压缩壳和加密壳。压缩壳出现较早，可追溯到DOS时代，使用压缩壳可以帮助缩减PE文件大小，隐藏PE文件内部代码和资源，便于网络传输和隐藏。压缩壳通常有两种用途，一种是单纯用于压缩普通的PE文件，另一种则会对源文件产生较大的变形，严重破坏PE文件头，通常用于压缩恶意程序。常见的压缩壳有：Upx、ASpack、PECompat。加密壳，也称为保护壳，它主要的功能是防止逆向分析技术，保护PE文件不被逆向分析。加密壳保护的文件通常比PE源文件大得多。常见的加密壳有：ASProtector、Armadillo、 EXECryptor、Themida、VMProtect。

现阶段主要的查壳手段有两种：基于特征码和基于信息熵。

基于特征码：同样的加壳方法会使得加壳后的PE文件在特定的位置有相同的字节序列。相同的序列即为该加壳方式的特征，然而这种基于特征码的查壳方法，只能检测出已有的并且加入到特征库的加壳方法。

基于信息熵：经过加壳的PE文件，其结构会产生变化。壳一般分为加密壳和压缩壳，加密和压缩会使 PE文件变成随机性更大的无结构形式。然而，熵是用来衡量不确定性的指标，这样经过加壳的PE文件的熵一般会大于源PE文件的熵。

目前有很多加壳工具，常用的有ASPACK、PE PACK、UPX、PECOMPACT 等。相应的也有一些查壳软件有PEID、FILEINFO、FILE SCANNER等，本文主要是用PEID的批量模式查壳。

3.3.2 反汇编技术

反汇编指将机器代码转换为汇编代码、低级转高级的意思，常用于逆向工程领域。目前网络上许多“免费软件”，PSP、PS、NDS游戏机的破解和苹果系统的越狱都跟反汇编息息相关。

基本的反汇编算法有两种：线性扫描反汇编和递归下降反汇编。

线性扫描反汇编算法从程序的机器指令进行全盘扫描，从程序的第一个指令字节开始，对所有检测到的指令以线性模式逐条反汇编，顺序的把所有字节码转换成指令，直到遇到程序的终结点。

递归下降反汇编算法重视控制流的概念，该算法按照指令是否为转移指令决定下一步操作。将整个过程分为两类，要么顺序遍历，要么跳转到另一个程序模块。当遇到指令转移指令时，将此位置作为一个分岔点，然后顺序反汇编每一个分支，每一个分支中如果又有分岔点，做同样的递归处理，直到遇到程序的终结点。

反汇编工具有很多，如有IDA Pro、C32Asm、W32DASM、花指令清除器1.2 等等。本文使用objdump工具对恶意代码样本进行批量反汇编，主要使用objdump -d指令，配合PowerShell脚本完成恶意代码样本的反汇编。

3.4 概率矩阵的生成

3.4.1 操作码抽象化

根据不同的抽象方式对汇编操作码做抽象化处理，得到各个中间码序列，以此来尝试从不同视角来理解程序。

本次实验会使用两种抽象化方式，分别记为Abs1和Abs2，其中Abs1是作者根据对汇编码的理解提出的抽象方式，中间码种类为9，Abs2是文献[1]中提出的抽象化方式，中间码种类为5。下面分别对Abs1和Abs2进行详细介绍。

Abs1将汇编操作码抽象为9类中间码：

（1）数据传输，比如：MOV、PUSH、POP、IN、OUT等；

（2）算术运算，比如：ADD、SUB、CMP、DIV、MUL等；

（3）逻辑运算，比如：NOT、AND、OR、TEST等；

（4）移位运算，比如：SAL、SHL、SAR、SHR等；

（5）字符串操作，比如：MOVS、STOS、CMPS等；

（6）过程控制，比如：JMP、JZ、JP等；

（7）标志位控制，比如：CLC、STD、STC等；

（8）CPU控制，比如：HLT、WAIT、NOP等；

（9）其他，伪指令及不常用的指令，比如：SEGMENT，ASSUME，END等。

这9种中间码的表示方法如表3-1：

表3-1 Abs1抽象中间码的表示

种类表示方法

数据传输 DATA_TRANS

算术运算 ALTH_OPE

逻辑运算 LOGIC

移位运算 SHIFT

字符串操作 STR_OPE

过程控制 PRO_CTRL

标志位控制 FLAG_CTRL

CPU控制 CPU_CTRL

其他 OTHER

Abs2将汇编操作码抽象为5类中间码：

（1）数据传输，比如：MOV，PUSH，IN等；

（2）运算，其中包含算数运算指令(ADD、SBB等)和逻辑运算指令(OR，XOR，TEST等)；

（3）程序转移，这是指影响程序正常流程的指令，比如：JMP，CALL，RET，INT等；

（4）处理机控制，比如：NOP，HLT，WAIT等；

（5）其他，伪指令及不常用的指令，比如：SEGMENT，ASSUME，END等。

这5种中间码的表示方法如表3-2：

表3-2 Abs2抽象中间码的表示

种类表示方法

数据传输 ASSIGN

运算 TEST

程序转移 JMP

处理机控制 CTRL

其他 OTHER

抽象化过程的实现流程如下：

（1）首先将所有的汇编操作码与中间码映射关系存储在配置文件中，配置文件格式如图3-3。其中IC（Intermediate Code）代表中间码，Abs1代表抽象名称，asm_code代表汇编操作码；

（2）循环读取配置文件的每一行。符号'#'表示忽略其后面的内容，符号'@'表示其后为抽象化方式的名称，从下行开始直到结束或者遇到符号'@'，中间的内容都是该抽象方式的具体内容，符号'：'前表示中间码名称，符号'：'后面的内容以符号'，'分隔，每一个都表示一种汇编操作码；然后将读取到的内容存储到类型map<string,map<string,string>>中，三个string类型分别表示抽象方式名称，汇编操作码、中间码。

图3-3 抽象化配置格式

（3）读取汇编文本文件，遍历其中的每一个汇编操作码，对其应用每一种抽象方式，得到对应的中间码表示，最后得到N种中间码序列，N表示配置文件中抽象方式的个数。

3.4.2 n-gram算法提取特征

n-gram是计算机语言学和概率论范畴内的概念，目前n-gram被广泛应用于自然语言的自动分类功能。该模型基于第n个词出现只和其前n-1个词相关的假设，对于处理依靠序列关系分类的问题具有很强的优势。

为避免多次读写文件影响系统性能，本文在得到中间码序列之后，继续在内存中使用n-gram算法得到中间码特征序列，并将中间码特征序列写入文件中。具体算法描述如表3-3：

表3-3 n-gram算法获取中间码特征序列描述

算法1：应用n-gram算法获取中间码特征序列

Input: IC_sequenceOutput: n-gram sequenceindex=start_indexfrom index to size(IC_sequence)-1{if index+n < size(IC_sequence){ret.push_back(<IC_sequence[index:index+n-1]>)}else{break;}}return ret

（1）设置当前索引为起始位置；

（2）判断当前位置是否合法，如果合法，转步骤（3）；否则转步骤（4）；

（3）判断当前位置索引加上n值是否合法，如果合法，则将当前位置开始往后的n-1个序列作为一个整体加入结果集，并转步骤（2）；否则，转步骤（4）；

（4）返回结果集并结束算法。

该算法结束时，将会得到一个n-gram中间码特征序列，每一个序列依次对应一个恶意代码样本。本文将同一类别的恶意代码的中间码特征序列放在同一个文件中，每一个序列占用一行。

3.4.3 特征分析

对中间码应用n-gram算法提取特征后，对于本文，将会得到两种不同的中间码特征序列，本节将使用Eclat算法得到这两种中间码特征序列的频繁项集，并利用频繁项集得到中间码特征序列的平均相似度，然后选择平均相似度最小的一种中间码特征序列作为概率矩阵的生成依据。

本文使用Eclat算法来分析特征文本的频繁项集。尽管Apriori算法是最广为人知的关联规则挖掘算法，但是算法需要多遍扫描数据库因而会产生大量的候选项集，支持度的计算也很耗时。相对于Apriori算法，Eclat算法采用了等价类、深度优先遍历、求交集等策略，支持度计算效率有很大改善。

Eclat算法采用的数据结构分为两部分：项和事务。Eclat算法在遍历数据库时，将事务划分到项下，使得该算法相较于Apriori算法可以基于集合运算很方便的得到频繁项集。将该算法应用到本次实验，项对应n-gram特征序列，事务对应文本的行号。应用Eclat算法得到频繁项集的思路如下：

（1）首先生成项对应的事务集，如图3-4中左上角表格，第一列的ABCDE代表n-gram特征序列，第二列T1-T9代表行号；

（2）把所有项作为一个集合，并求该集合所有的子集，如图3-4中橘色线标出的集合；

（3）对子集中的所有项对应的事务集合求交集，作为该子集对应的事务集合Ti，如图3-4蓝色线指向的集合；

（4）如果集合Ti的基数大于设定的阈值，即判定该项为频繁项。

本文只统计单一的频繁项，并不会统计各个项组合的情况，因此本文对该算法做了简化，具体算法描述如表3-4：

（1）设置遍历起始行；

（2）循环遍历文件的每一行，分割该行字符串，将每一个特征序列作为key值，将行号插入到value所表示的集合中，直到文件结束；

（3）遍历步骤（2）统计到的结果，如果某一key值对应的value中存储的值的个数超过设定的阈值，则将该key值，即就是特征序列插入到结果集；直到循环结束返回结果集。

表3-4 Eclat算法描述

算法2：Eclat算法获取频繁项集

Input: n-gram IC_sequence fileOutput: Frequent Item Setsvector<string> n_gram_IC_sequences = read(n-gram IC_sequence file)index=start_indexfrom index to size(n_gram_IC_sequences)-1{vector<string>

items=split(n_gram_IC_sequences[index],’,’)from i = 0 to size(items)-1 {count[items[i]].insert(index)}}ret = getFrequentItems(count,Threshold)return ret

图3-4 Eclat算法示例图

在抽象方式一定的情况下，分别对每一个类别标签下的中间码特征序列应用算法2，得到各个类别在同一中抽象方式下的频繁项集，接着本文使用平均相似度评估该中间码特征序列生成的频繁项集对于恶意代码分类的效果，平均相似度越低，代表分类效果越明显，反之，代表分类效果差。频繁项集的平均相似度定义如式（3-1）：

AVGlike=cardA*α,βAα≠βcard(α∩β)Ccard(A)2*αAcard(α)（ 3-1）

式中α，β分别代表各类的频繁项的集合，A表示各类频繁项集的集合，card(A) 表示集合A的基数。例如，集合A={α,β,γ}，其中集合α频繁项为{a,b,c,d}，集合β频繁项为 {a,b,e,f}，集合γ频繁项为{c,d,f,g}，那么

AVGlike=3*(cardα∩β+cardα∩γ+cardβ∩γ)C32*(4+4+4)=512≈41.67%

得到了各中间码特征序列生成频繁项集的平均相似度之后，再从中选取平均相似度最小的中间码特征序列，作为生成概率矩阵的依据。

3.4.4 概率矩阵

应用特征分析步骤选择出的中间码特征序列，统计分析出其对应的概率矩阵，具体算法流程如表3-5所示。

（1）遍历选择的中间码特征序列文件，并获得每个词组在该行中出现的频率；

（2）对每一行循环N次，N代表n-gram算法中的n值，这将会组成该抽象方式下每一种可能出现的n-gram词组，记录该行出现这种形式词组的频率；

（3）将该行中间码特征序列频率分布作为代表该恶意代码的概率分布，继续对下一行做重复操作，直到文件结束。

表3-5 获取概率矩阵的算法描述

算法3：应用中间码特征序列获取概率矩阵

Input: n-gram IC_sequence fileOutput: Probability matrix fileforeach line in n-gram IC_sequence file{line_pros = getWordProbabilityInLine(line)foreach IC_1 in AbsX{foreach IC_2 in AbsX{:foreach IC_N in AbsX{matrix_line += line_pros[“IC_1:IC_2:⋯:IC_N”])}} }writeFileLine(class，matrix_line)clear(matrix_line)}

最终本文将会获得一个含有类别标签的概率矩阵文件，用于机器学习算法的模型训练以及评估。至此，特征提取部分全部完成。

3.5 恶意代码分类

3.5.1 随机森林算法

随机森林是一种集成学习算法，该算法在学习过程中将产生多个决策树，每棵决策树会根据输入数据集产生相应的预测输出，算法采用投票机制选择类别众数作为预测结果。随机森林的训练模型生成流程如图3-5所示。

图3-5 随机森林算法训练模型生成过程

首先采用bootstrap对数据进行采样，样本总数为N，那么每次也随机采样N个样本作为单个决策树的训练数据集，采样是有放回的采样，所以并不是使用样本空间中的每一个样本作为单个决策树的训练集数据。在每个节点，算法首先随机选取m（m<<M）个变量，从它们中间找到能够提供最佳分割效果的预测属性；然后，算法在不剪枝的前提下生成单个决策树；最后从每棵决策树都得到一个分类预测结果。如果是回归分析，算法将所有预测的平均值或者加权平均值作为最后输出，若果是分类问题，则选择类别预测众数作为最终预测。

本文通过实验对比分析，最终选择决策树的个数T为500，随机属性个数m为6。

3.5.2 SVM

很多研究已经证明支持向量机（Support Vector Machine，SVM）是一种强大的分类工具，可以被广泛的应用到不同的领域。与随机森林算法不同，在SVM训练中，从输入数据到输出结果的过程并不清晰，也难以解释，因此SVM属于黑盒算法。SVM算法的基本思想：通过定义核函数将输出数据映射到高维特征空间上，并在此空间中构造一个最有分类超平面（或者一组超平面），使得高维特征空间内的各个类的边缘间隔最大化。定义这些超平面的向量就被称为支持向量，如图3-7中被正方形圈出的o和x：

图3-6 SVM分类示意图

SVM算法的关键在于核函数，常用的核函数主要有四种：

（1）线性核函数：Kx,y=x·y;

（2）径向基函数：Kx,y=exp-γx-y2,γ>0;

（3）多项式核函数：Kx,y=γx+y+rd,γ>0;

（4）二层神经网络核函数：Kx,y=tanhγx·y+r;

式中的γ，r和d都是核参数。

SVM算法是最初设计是为解决二分类问题的，但是本此实验是一个多分类问题，所以需要对SVM算法做一些改动。目前针对SVM多分类问题主要有两种解决方案：直接法和间接法。

直接法：即直接修改目标函数，但是这种算法复杂度较高，很难实现，只适合解决规模较小的问题。

间接法：通过组合多个二分类器实现多分类器功能，常见的方法有两种：一对一法和一对多法。一对多法可能会出现训练

集分布不均的情况，甚至可能会有未分类的情况出现，所以本文使用一对一的方法，其算法描述如表3-6所示。对每一个类别的训练集，都和类别集合中的其他类别训练集使用SVM算法构建二分类模型，如果类别总数为n时，二分类SVM算法将会执行n+1*n2次，接着待检测样本使用每一个二分类模型进行预测，最后选取最频繁的预测值作为总的预测结果。这也是libsvm中SVM多分类的实现的方式。

表3-6 SVM实现多分类算法描述

算法4：应用SVM实现多分类

Input：multi-classes trainsetOutput：sample classification resultforeach class_i in classeset{foreach class_j in classeset except class_i{multi_classification_svm_model.insert(svm_model(class_i_trainset, class_j_trainset))}}return getMostFrequentClass(predict(sample,multi_classification_svm_model))

3.5.3 KNN

K近邻（K-Nearest Neighbor，KNN）算法属于一种无参惰性学习方法。无参类算法不会对数据的分布做任何的假设，而惰性学习方法则不要求算法具备显性学习过程。算法的主要思想是：通过对样本进行建模，使得可以用一个向量表示一个样本，然后采用一种度量方式，度量方式一般使用欧式距离或者曼哈顿距离，计算出离被测样本最近的K个已知类别的样本，接着统计这K个样本的类别次数，则判定被测样本的类别与出现次数最多的类别相同，为了避免出现平票的情况，K一般选择为奇数。

KNN算法步骤如下：

（1）首先对特征向量进行归一化处理，本文采用线性函数的转换方法：

y=(x-min)/(max-min) (3-2)

其中y为归一化处理之后的值，x为处理前的值，max和min分别代表一个样本中出现的最大值和最小值。

（2）使用一种度量方式计算待检测恶意代码样本和所有已知类别数据样本的距离，本文使用欧式距离；

i=1k(xi-yi)2（3-3）

（3）按照距离递增进行排序；

（4）选取和当前待测恶意代码样本距离最小的K个点；

（5）确定前K个点所属类别的出现频率；

（6）返回前K个点出现频率最高的类别作为当前点的预测类别，则目标函数为：

fv=argmaxc∈Ci=1kδ(c,f(vi)) (3-4)

其中，函数fv表示特征向量v的类别，如果f(vi)的类别和c相同，则δ(c,fvi=1，否则δ(c,fvi=0。

本文使用KNN算法来做实验仿真分析，通过实验选择分类较好的K值为5。

3.6 实验仿真分析

3.6.1 实验评价方法

为了确保模型能对未知或新到达的对象进行正确预测，需要对模型性能进行评估，避免模型可能存在过拟合问题。

k折交叉验证技术能够解决过拟合问题，因此被广泛应用于分类器性能评估领域。k折交叉验证并不需要使用整个数据集，相反，它会将数据集划分为训练集和测试集两部分。这样，基于训练集得到的模型就可以通过测试集来完成性能测评。重复执行n次k折交叉验证后，就能够根据n次检验的平均正确率实现对模型的真实评估。本次实验n取值3，k取值为10，即就是3重10折交叉验证。

对于恶意代码的检测系统，一般采用准确率（Accuracy），精确率（Precision）和召回率（Recall）来评价其结果。用TP表示恶意样本被正确分类的数量；FN表示恶意样本被判定为正常样本的数量；TN表示正常样本被正确分类的数量；FP表示正常样本被判定为恶意样本的数量。

指　标
疑似剽窃文字表述

1. 首先是样本预处理，该步骤包括对所有的样本进行查壳脱壳处理，并对脱壳后的样本进行反汇编，得到其汇编文本文件；其次，
2. 计算机软件中一段专门保护软件不被非法修改或反编译的程序。它们一般都先于程序运行，拿到控制权，然后完成保护软件的功能。
3. 现阶段主要的查壳手段有两种：基于特征码和基于信息熵。
   基于特征码：同样的加壳方法会使得加壳后的PE文件在特定的位置有相同的字节序列。相同的序列即为该加壳方式的特征，然而这种基于特征码的查壳方法，只能检测出已有的并且加入到特征库的加壳方法。
   基于信息熵：经过加壳的PE文件，其结构会产生变化。壳一般分为加密壳和压缩壳，加密和压缩会使 PE文件变成随机性更大的无结构形式。然而，熵是用来衡量不确定性的指标，这样经过加壳的PE文件的熵一般会大于源PE文件的熵。
   目前有很多加壳工具，常用的有ASPACK、PE PACK、UPX、PECOMPACT 等。相应的也有一些查壳软件有PEID、FILEINFO、FILE SCANNER等，本文主要是用PEID的批量模式查壳。

4. 基本的反汇编算法有两种：线性扫描反汇编和递归下降反汇编。
   线性扫描反汇编算法从程序的
5. 汇编工具有很多，如有IDA Pro、C32Asm、W32DASM、花指令清除器1.2 等等。本文使用
6. n-gram被广泛应用于自然语言的自动分类功能。该模型基于第n个词出现只和其前n-1个词相关的假设，对于处理依靠序列关系分类的问题具有很强的优势。
7. 计算出离被测样本最近的K个已知类别的样本，接着统计这K个样本的类别次数，则判定被测样本的类别与出现次数最多的类别相同，
8. 用TP表示恶意样本被正确分类的数量；FN表示恶意样本被判定为正常样本的数量；TN表示正常样本被正确分类的数量；FP表示正常样本被判定为恶意样本的数量。

5. 第三章基于机器学习算法的恶意代码检测方法_第2部分

总字数：3546

相似文献列表　文字复制比：0%(0)　疑似剽窃观点：(0)

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

对于多元分类模型，准确率、精确率和召回率的计算公式如下，其中l表示分类个数：

Average Accuracy=1li=1lTPi+TNiN（3-5）

Average Precision=1li=1lTPiTPi+FPi（3-6）

Average Recall=1li=1lTPiTPi+FNi（3-7）

3.6.2 实验环境与数据

实验环境：Intel酷睿i5-7300HQ@2.50GHz 双核，8.00G内存，操作系统为Microsoft Windows 10，编程语言主要是C++。本文算法实现采用C++语言编程，利用CodeBlocks集成工具实现相关算法的程序设计。

本文使用的实验数据来自Kaggle上微软发起的一个恶意代码分类的比赛，使用的数据集的大小为136GB，其中样本种类分布如表3-7：

表3-7 恶意程序数据集

恶意程序家族	数量（单位，个）
Ramnit	1541
Lollipop	2478
Kelihos_ver3	2942
Vundo	475
Simda	42
Tracur	751
Kelihos_ver1	398
Obfuscator.ACY	1228
Gatak	1013
total	10868

3.6.3 特征分析与实验结果

应用3.4.3小节特征分析中提到的算法，得到特征分析的结果如图3-7所示, 由图3-7可以看出，对于2-gram算法，Abs1对应的中间码特征序列得到的频繁项集平均相似度较小，本文改进的方法将会选择该中间码特征序列生成概率矩阵；而在3-gram和4-gram算法中，Abs2对应生成的中间码特征序列得到的频繁项集平均相似度较低，本文方法将会选择该中间码特征序列生成概率矩阵。

为了证明特征分析阶段选择生成的概率矩阵确实会得到更好的结果，本次实验将Abs1以及Abs2对应的概率矩阵都生成，并且使用机器学习算法来对样本进行分类，对比分类效果。同时为了分析出哪种机器学习算法使用本文生成的概率矩阵分类效果最佳，本次实验将会分别使用随机森林算法、SVM和KNN算法对实验进行分类。

图3-7 平均相似度对比图

下面对实验结果进行分析。

1. 准确率比较

下面通过对3种n-gram、2种抽象方式和3种机器学习算法的18种不同的组合来对比准确率。

图3-8中，横坐标表示不同的机器学习算法，纵坐标为准确率。其中蓝色柱状抽象方式Abs1，橘色柱状表示抽象方式Abs2，RF代表随机森林算法，并且下面的图中均表达此种意义。

对比图3-8的结果，当n=2时可得出如下结论：

图3-8 n=2时准确率对比图

（1）平均准确率：SVM>随机森林算法>KNN。

（2）算法稳定性：随机森林算法>KNN>SVM。

（3）是否符合特征分析结果：符合，不论哪种机器学习算法，Abs1对应的准确率始终高于Abs2对应的准确率，所以实验结果符合特征分析判断的结果。

对比图3-9的结果，当n=3时可以得出如下结论：

图3-9 n=3时准确率对比图

（1）平均准确率：随机森林算法>KNN>SVM。

（2）算法稳定性：随机森林算法>KNN>SVM。

（3）是否符合特征分析结果：符合，不论何种机器学习算法，抽象方式为Abs2时，准确率均高于抽象方式为Abs1时的准确率。

对比图3-10的结果，当n=4时可以得到如下结论：

（1）平均准确率：KNN>随机森林算法>SVM。

（2）算法稳定性：KNN>随机森林算法>SVM。

图3-10 n=4时准确率对比图

（3）是否符合特征分析结果：符合，不论采用何种机器学习算法，抽象方式为Abs2时，其准确率均大于抽象方式为Abs1时的准确率。

2. 精确率比较

下面通过对3种n-gram、2种抽象方式和3种机器学习算法的18种不同的组合来对比精确率。

图3-11中，横坐标表示不同的机器学习算法，纵坐标为精确率。其中蓝色柱状表示抽象方式Abs1，橘色柱状表示抽象方式Abs2，RF代表随机森林算法，并且下面的图中均表达此种意义。下面通过不同的n值和不同的机器学习算法来对比精确率。

图3-11 n=2时精确率对比图

对比图3-11的结果，当n=2时可得出如下结论：

（1）平均精确率：SVM>随机森林算法>KNN。

（2）算法稳定性：KNN>随机森林算法>SVM。

（3）是否符合特征分析结果：符合，无论采用何种机器学习算法，抽象方式为Abs1时，精确率都大于抽象方式为Abs2的情况。

对比图3-12的结果，当n=3时可以得出如下结论：

（1）平均精确率：随机森林算法>KNN>SVM。

（2）算法稳定性：随机森林算法>KNN> SVM。

（3）是否符合特征分析结果：比较不符合，除了SVM算法的精确率在Abs2时大于Abs1，随机森林算法和KNN算法都不符合特征分析结果。

图3-12 n=3时精确率对比图

对比图3-13的结果，当n=4时可以得出如下结论：

（1）平均精确率：KNN>SVM>随机森林算法。

（2）算法稳定性：KNN>随机森林算法> SVM。

（3）是否符合特征分析结果：符合，抽象方式为Abs2时，精确率都大于抽象方式为Abs1的情况。

图3-13 n=4时精确率对比图

3. 召回率比较

下面通过对3种n-gram、2种抽象方式和3种机器学习算法的18种不同的组合来对比召回率。

图3-14中，横坐标表示不同的机器学习算法，纵坐标为召回率。其中蓝色柱状表示抽象方式Abs1，橘色柱状表示抽象方式Abs2，RF代表随机森林算法，并且下面的图中均表达此种意义。下面通过不同的n值和不同的机器学习算法来对比召回率。

对比图3-14的结果，当n=2时可得出如下结论：

图3-14 n=2时召回率对比图

（1）平均召回率：随机森林算法>SVM> KNN。

（2）算法稳定性：随机森林算法>KNN> SVM。

（3）是否符合特征分析结果：基本符合，除了KNN算法在Abs1时召回率小于Abs2的情况，其他两种机器学习算法都在Abs2时召回率大于Abs1。

对比图3-15的结果，当n=3时可得出如下结论：

（1）平均召回率：随机森林算法> KNN>SVM。

（2）算法稳定性：随机森林算法>KNN> SVM。

图3-15 n=3时召回率对比图

（3）是否符合特征分析结果：符合，三种机器学习算法在Abs2时召回率都大于Abs1时的召回率。

对比图3-16的结果，当n=4时可以得出如下结论：

图3-16 n=4时召回率对比图

（1）平均召回率：随机森林算法> KNN>SVM。

（2）算法稳定性：随机森林算法>KNN> SVM。

（3）是否符合特征分析结果：符合，三种机器学习算法在Abs2时召回率都大于Abs1时的召回率。

4. 实验结果分析

通过对实验结果的对比分析，可以得出如下结论：

（1）本文方法相比于传统方法有一定的优越性。在2-gram算法中，本文方法将会选择Abs1对应的中间码特征序列生成概率矩阵，而传统的方式只能使用Abs2的中间码特征序列，图3-17显示了本文方法和传统方法在准确率、精确率和召回率指标上的差距，可以看出本文与传统方法相比会得到更好的结果，并且在2-gram时这三个指标的平均值是18种组合里最高的；对于3-gram算法和4-gram算法，本文选择的抽象方式Abs2，与传统方法相同，都会得到准确率、精确率和召回率较高的结果。

图3-17 n=2时本文方法和传统方法对比图

（2）随机森林算法是三种算法中稳定性最好的，并且在准确率、精确率和召回率三个指标中的结果比较令人满意，所以在要求平均性能较好的情况下，优先选择随机森林算法。SVM算法是最有可能出现峰值的算法，但是也是相对来说最不稳定的一个算法。相比而言，KNN算法是最不出众的算法，虽然稳定性优于SVM算法，但是其平均性能不如随机森林算法。所以本文实现的机器学习算法的恶意代码检测系统将会选择随机森林算法来进行检测。

3.7本章小结

本章首先分析了汇编操作码的特点，在分析了相关文献之后，改进了一种基于机器学习的恶意代码检测算法；接着简要介绍了该算法的主要流程，然后对算法的各步骤进行了详细的说明，重点对概率矩阵的生成进行了详尽的解释说明，然后简单介绍了机器学习算法、SVM算法和KNN算法的基本原理和在本实验中的使用情况；最后对实验仿真结果进行了分析，并给出结论。

6. 第四章系统设计与实现

总字数：2834

相似文献列表　文字复制比：1.2%(33)　疑似剽窃观点：(0)

1	201091303535322 -《学术论文联合比对库》- 2013-06-24	1.2%（33） 是否引证：否

原文内容　红色文字表示存在文字复制现象的内容; 绿色文字表示其中标明了引用的内容

第四章系统设计与实现

4.1 系统设计

本文实现的系统主要目的是：用户能够使用一个相对友好的界面去提交想要检测的程序，系统接收到程序之后，能够快速而准确向用户反馈检测结果。

系统用例图如图4-1所示：

图4-1 恶意代码检测系统用例图

选择上传程序功能：用户上传需要检测的本地程序，现阶段系统仅支持PE文件，即格式为EXE、DLL、OCX、SYS和COM格式的文件。上传程序之前会对文件格式进行检测，如果用户上传文件格式不符合要求，则拒绝上传并提示用户重新选择文件进行上传。

查看检测结果功能：用户上传了需要检测程序后，系统后台会返回检测结果，并由本地展示检测结果。

反馈结果：当用户认为结果与实际情况不符或者检测结果不可信时，可向系统替提交反馈，以便系统进行再次确认。

针对上述用例图，本文提出了一种系统的设计框架，如图4-2所示：

图4-2 系统框架图

系统共包含5个模块：UI模块，通信模块，检测模块，存储模块和模型训练模块。

UI模块主要有两个功能：上传待检测程序和展示检测结果。

通信模块主要由两个功能：对用户上传的程序进行存储和以及向用户传输检测结果。

检测模块负责对用户提交的程序给出检测结果。当得到用户上传的程序之后，对程序做MD5运算得到哈希码，首先使用该哈希码去查询数据库，如果在数据库中查询成功，则表示该程序已经被检测过，直接返回检测结果；否则使用系统的检测模型，对程序进行检测，得到分类结果之后，将结果交给通信模块向用户报告结果，并且将分类结果交给存储模块进行存储。

存储模块负责对检测结果进行存储并负责向模型训练模块提供训练样本。

模型训练模块负责定时或者主动的对存储模块中的数据进行重新训练，更新检测模型。

4.2 主要功能模块

4.2.1 UI模块

UI模块功能比较简单，为了方便恶意代码的检测，本文使用了CS架构实现检测系统，用户只需在网页上传需要检测的程序，然后可以在本地查看检测结果。

开始界面展示如图4-3所示，选择上传需要检测的文件界面如图4-4。

图4-3 web端首页展示

图4-4 上传程序界面

上传文件之后，点击开始检测，得到检测结果如图4-5所示：

图4-5 检测结果展示图

4.2.2 通信模块

通信模块是UI模块和后台模块交互的唯一接口，通信模块主要有以下功能：

（1）保证客户端和服务端的正常交互。

（2）接收客户端上传的程序文件或者反馈信息，并通知相应模块。

（3）将检测模块的分析结果返回给用户。

通信模块主要使用tomcat7.0.82完成。

4.2.3 检测模块

检测模块是系统的核心模块，该模块接收来自通信模块的消息，当被告知有程序需要检测，检测模块首先会对程序求MD5值，然后将该MD5值传给存储模块，由存储模块先在数据库中去查找该MD5，如果查找成功，则返回存储的检测结果给检测模块，由检测模块将检测结果返回给通信模块；如果在数据库中查找失败，则使用系统中现有的预测模型对该程序进行检测。检测完成之后，首先将结果传输给存储模块进行存储，再将结果交由通信模块。具体过程如图4-6所示。

当数据库中不存在待检测程序的MD5值时，程序将会根据系统已有的检测模型对程序进行检测。具体过程如图4-7所示。首先，对程序进行预处理，包括查壳和脱壳以及反汇编过程；然后从汇编文件中提取汇编码，并根据现有预测模型的抽象方式和n-gram中n的取值提取程序的特征码序列，并构建概率矩阵，图中AbsX表示抽象方式编号为X；最后使用概率矩阵作为预测模型的输入，得出该程序的检测结果。

图4-6 检测模块时序图

图4-7 未知恶意程序检测过程

4.2.4 存储模块

存储模块主要使用MySQL数据库对PE文件的检测结果进行存储，数据库的设计也比较简单，目前数据库中存在两个表Tb_Malicious_Detected和Tb_Types，Tb_Malicious_Detected表存储所有被检测过的程序及其检测结果，表Tb_Types存储本系统所能识别的所有类别。表Tb_Malicious_Detected结构如图4-8所示：程序的MD5值作为主键，存储类型为32位的char类型，detect_time表示检测时间，if_feedback表示检测结果是否被用户反馈为结果不可信，取值为0或者1，0表示未反馈，1表示反馈，默认值为0，detect_result表示检测结果，存储类型为int类型。将具体类型名称和类型编号对应关系存储在表Tb_Types中，在表Tb_Malicious_Detected中不直接存储类型名称的原因是：随着系统的使用，数据必然会越来越多，将会存在大量的同一类型的程序，这个字段如果存储为字符串类型，会占用大量的磁盘空间。表Tb_Types格式如图4-9所示，自增id为主键，int_type表示程序类型的整型表示，str_type表示程序类型的字符串表示，不直接使用int_type为主键主要考虑到今后可能会对程序类型的划分方式做调整，可以对int_type进行不同解释达到这个目的。目前可识别的类型共有10种，包括3.6.2小节种提到的的9种恶意代码类型和1中正常类型。

图4-8 表Tb_Malicious_Detected结构

图4-9表Tb_Types结构

4.2.5 模型训练模块

随着系统的使用，数据量必然会增大，如何利用这些不断增长的数据去使得系统的检测性能更好，这就需要系统不定时的利用已有数据去主动学习，不断改进系统的检测模型。模型训练模块和第三章所述的过程完全一样，只不过数据的标签需要从数据库中获取。

模型训练模块时序图如图4-10所示。首先从数据库中读取所有的标签数据，然后从文件系统中读取上一次构建模型时生成的n-gram中间码序列，本系统中共有6个文件：2-gramAbs1、2-gramAbs2、3-gramAbs1、3-gramAbs2、4-gramAbs1和4-gramAbs2；接着将文件系统中指定位置存储的PE文件进行预处理、抽象化并使用n-gram算法得到6种不同的中间码特征，将其加入到各自对应的中间码特征文件中，此时，需要将6个中间码特征序列文件存储更新到文件系统，并选择删除或者移动所有的PE文件，防止下次进行模型训练时重复被使用；再对这6个文件进行特征分析，选择出频繁项集的平均相似度最小的一种，依据该中间码特征序列生成概率矩阵，最后使用随机森林算法得到检测模型，并存储检测模型、n-gram算法n值和对应的抽象方式，以便检测模块使用。

图 4-10 模型训练模块时序图

4.3 本章小结

本章主要介绍了基于机器学习算法的恶意代码检测系统的设计与实现。先概括介绍了该系统的整个设计框架；然后对各个模块的功能、设计思路和实现方式分别进行了分析讨论。

第五章总结与展望

5.1 本文工作总结

计算机的普及和高速发展的互联网技术带给人们方便的同时，用户信息的安全性受到的威胁也越来越严重，加强互联网安全已经成为了国家的一项重要战略。在众多的网络安全事件中，尤以恶意代码的危害最大，给个人、企业甚至政府造成了巨大的损失，所以恶意代码检测技术成为了信息安全领域重要的研究方向。本文主要研究了基于机器学习算法的恶意代码检测技术，介绍了信息安全和机器学习领域的相关理论，分析了恶意代码检测技术的国内外研究现状，在研究分析了大量恶意代码以及相关机器学习算法之后，改进了一种基于机器学习算法的恶意代码检测方法，通过实验证明了本文方法对恶意代码的分类具有明显的效果，并依据该方法设计实现了一个恶意代码检测系统。

本文完成的主要工作有：

（1）对实验样本进行收集和预处理操作，研究了相关的恶意代码检测分析技术。

（2）改进了一种基于机器学习算法的恶意代码检测方法，主要是使用汇编操作码的抽象化技术，并结合数据挖掘技术选择合适的抽象方式对应的中间码特征序列生成概率矩阵，作为代表恶意代码的特征。当n-gram算法中n值有多个时，这种方法能够针对特定的n值，选择合适的中间码特征序列作为概率矩阵的依据，并由此概率矩阵构建效果比较好的分类模型。通过实验分析，本文改进的方法具有一定的优越性。在三类机器学习算法中，随机森林算法针对于本文方法生成的概率矩阵，能够训练出综合性能相对较好的分类模型。从整个实验结果分析，本文改进的方法具有较高的准确率、精确率和召回率。

（3）设计并实现了一个恶意代码检测系统，采用模块化编程实现系统，增加了系统的灵活性。

5.2 未来展望

本文虽然改进了一种基于机器学习算法的恶意代码检测方法，并且基于该方法实现了一个恶意代码检测系统，但由于时间及各方面条件的限制，本文改进的方法还未在大规模实践中得到验证，这也是下一步着重需要解决的问题。同时在基于机器学习算法的恶意代码检测领域中还有一些问题需要进一步的研究解决。

（1）恶意代码种类层出不穷，其反检测的方法也多种多样，保证得到正确的汇编操作码是恶意代码静态分析的前提。如何有效的对抗反检测技术是未来的一个研究方向。

（2）本文提到的概率矩阵，有一定的适用范围，比如当中间码类别数目为c时，概率矩阵的列数是cn，指数n表示n-gram中n的取值，随着n的增大，概率矩阵列数呈指数级增长，并且对于大多数的列，其值为0。研究如何有效的存储概率矩阵并针对稀疏矩阵做出优化需要后续的研究。

（3）对于汇编操作码的抽象方式，并没有过多的参考文献可供参考，如何对汇编操作码进行有效的抽象化处理也需要进一步的研究。

（4）伴随着数据集的急剧增长，算法的运行时间也在不断增加，对于该算法如何应用在分布式计算框架中以解决算法计算效率问题，也需要在未来的研究中不断探索。

参考文献

[1] 王冰，方勇. 基于汇编指令分布的恶意代码检测算法研究[J]. 信息安全研究. 2015.12:267-271.

[2] Ed S, Lenny Z.决战恶意代码[M]. 北京: 电子工业出版社 2005.展, 2005:1-10.

[3] Sung A H, Xu J, Chavez P, et al. Static Analyzer of Vicious Executables[C]. Proceedings of the 20th Annal Computer Security Applications Conference, Tucson, AZ, USA. IEEE Computer Society Press, 2004:326-334.

[4] Zhang B Y, Yin J P, Tang W S, et al. Unknown Malicious Codes Detection Based on Rough Set Theory and Support Vector Machine[C]. 2006 International Joint Conference on Neural Networks Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada. July 2006:16-21.

[5] Trevor H, Peter S, Harald S. State Joining and Splitting for the Symbolic Execution of Binaries[J], Lecture Notes in Computer Science. 2009, 5779:76-92.

[6] Edmund M C, Doron A P, Model Checking[M]. MIT Press, 1999:5-23.

[7] Patrick C, Abstract Interpretation Based Formal Methods and Future Challenges[C]. In Informatics, 10 Years Back — 10 Years Ahead, R. Wilhelm (Ed.), Lecture Notes in Computer Science 2000. Springer, 2001:138-156.

[8] Christodorescu M, Jha S. Static Analysis of Executables to Detect Malicious Patterns[C]. In Proceedings of the 12th Conference on Usenix Security Symposium, Washington DC, USA, 2003. USENIX Association: 12-32.

[9] Christodorescu M, Jha S, Seshia S. Semantics-Aware Malware Detection[C]. 2005 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2005.Institute of Electrical and Electronics Engineers Inc, 2005:32-46.

[10] Preda M D, Giacobazzi R. Control Code Obfuscation by Abstract Interpretation[C]. The Third IEEE International Conference on Software Engineering and Formal Methods (SEFM '05), Koblenz, Germany, 2005. IEEE Computer Society: 301-310.

[11] Singh P, Lakhotia A. Static Verification of Worm and Virus Behavior in Binary Executables Using Model Checking[C]. Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society, West Point, New York, USA. IEEE: 298-315.

[12] Kinder J, Katzenbeisser S, Schallhart C, et al. Detecting Malicious Code by Model Checking[C]. In Proceedings of the Second International Conference on Detection of Intrusions and Malwares, and Vulnerability Assessment (DIMVA'05), Vienna, Austria, 2005. Springer Verlag:174-187.

[13] 李佳静，梁知音，韦韬，毛剑. 一种基于语义的恶意行为分析方法[J]. 北京大学学报(自然科学版). 2008, 44(4):538-542.

[14] 王晓洁，王海峰. 基于语义的恶意代码检测的算法研究[J]. 计算机系统应用. 2009, 8:103-106.

[15] 孔德光. 结合语义的统计机器学习方法在代码安全中应用研究[D]. 2010:1-60.

[16] Tahan G, Glezer C, Elovici Y，et al. Auto-Sign: an Automatic Signature Generator for High-speed Malware Filtering Devices[J]. Comput Virol 2010 , 6:91-103.

[17] Tang Y, Xiao B, Lu X C. Using a Bioinformatics Approach to Generate Accurate Exploit-based Signatures for Polymorphic Worms[C]. Computers & Security 2009, 28:827-842.

[18] Wang L J, Li Z C, Chen Y, et al. Thwarting Zero-Day Polymorphic Worms With Network-Level Length-Based Signature Generation[C]. IEEE/ACM TRANSACTIONS ON NETWORKING.VOL.18.NO.1.FEBRUARY 2010: 53-65.

[19] Robert M, Clint F, Eugene B, et al. Unknown Malcode Detection Using OPCODE Representation[C]. In Proceedings of the 1st European Conference on Intelligence and Security Informatics (EuroISI). 2008:204-215.

[20] Chawla N, Japkowicz N, Kotcz A. Editorial: Special Issue on Learning from Imbalanced Data Sets[J]. SIGKDD Exploratuons Newsletter, 2004, 6(1):1-6.

[21] Schultz M, Eskin E, Zadok F, et al. Data Mining Methods for Detection of New Malicious Executables[C]. In Proceedings of the 2001 IEEE Symposium on Security and Privacy, 2001:38-49.

[22] Kolter J, Maloof M. Learning to Detect and Classify Malicious Executables in the Wild[J]. Journal of Machine Learning Research, 2006, 7:2721-2744.

[23] Mitchell T. Machine Learning[M]. McGraw-Hill, New York, 1997:12-23.

[24] Henchiri Q, Japkowicz N. A Feature Selection and Evaluation Scheme for Computer Virus Detection[C]. In: Proceedings of ICDM 2006, Hong Kong. 2006:891-895.

[25] Raja K S, Niklas L, Henric J. Accurate Adware Detection Using Opcode Sequence Extraction[C]. The Sixth International Conference on Availability, Reliability and Security. 2011:189-196.

[26] Dolev S, Tzachar N. Malware Signature Builder and Detection for Executable Code[C]. Patent Application. 2010:1-5.

[27] Sulaiman A, Ramamoorthy K, Mukkamala S, et al. Disassembled Code Analyzer for Malware (DCAM)[C]. In Proceedings of the IEEE International Conference on Information Reuse and Integration. 2005:398-403.

[28] Siddiqui M, Wang W, Lee J. Detection Internet Worms Using Data Mining Techniques[J]. Journal of Systemics, Cybernetics and Informatics. 2010, 6(6):48-53.

[29] Shahzad R K, Haider S I, Lavesson N. Detection of Spyware by Mining Executable Files[C]. In Proceedings of the International Conference on Availability, Reliability, and Security (ARES), 2010:295-302.

[30] Boojoong K, Hye S K, Taeguen K, et al. Fast Malware Family Detection Method Using Control Flow Graphs[C]. ACM, 2011:1-6.

[31] Ismail B, Aitor G. Graphs, Entropy and Grid Computing: Automatic Comparision of Malware[C]. In Proceedings of the Virus Bulletin Conference, 2008:54-61.

[32] Halvar F. Structural Comparison of Executable Objects[C]. In Proceedings of the IEEE Conference on Detection of Intrusions and Malware & Vulnerability Assessment, 2004:1-10.

[33] Igor S, Felix B, Xabier U P, et al. Opcode Sequences as Representation of Executables for Data-mining-based Unknown Malware Detection[J]. Inform.Sci, 2011:1-19.

[34] Perdisci R, Lanzi A, Lee W. Classification of Packed Executables for Accurate Computer Virus Detection[J]. Pattern Recognition Letters, 2008, 29:1941-1946.

[35] Perdisci R, Lanzi A, Lee W. McBoost: Boosting Scalability in Malware Collection and Analysis Using Statistical Classification of Executables[C]. In Proceedings of the 23rd Annual Computer Security Applications Conference, 2008:301-310.

[36] Hoglund G, McGraw G. Exploiting Software: How to Break Code[J]. US: Addison Wesley, 2004:60-81.

[37] 何永勇，诸福磊，钟乘林．基于进化计算的神经网络设计与实现[J]．,2001，16(3)．

[38] Schwartz E J，Avgerinos T，Brumley D．All you ever wanted to know about dynamic taint analysis and forward symb01ic execution(but might have been afraid to ask)[C]．Security and Privacy(SP)，2010 IEEE Symposiumon．IEEE．2010：317—331．

[39] 孙晓妍，祝跃飞，黄茜等．基于系统调用踪迹的恶意行为规范生成[J]．计算机应用，2010，30(7)：1767—1770．

[40] Kirda E，Kruegel C，Banks G，et a1． Behavior Based Spyware Detection [c]． Proceedings of the 15th USENIX Security Symposium．2006：19—19．

[41] A Danielescu，Anti-debugging and anti-emulation techniques，CodeBreakers Journal，2008
[42] Kranz J，Sepp A，Simon A. GDSL：A universal toolkit for giving semantics tO machine language[G] / / Programming Languages and Systems．Berlin：Spinger，2003：209—216.
[43] Alam S，Horspool R N，Traore I．MAIL：Malware analysis intermediate language一-A step towards automating and optimizing malware detection[C] / / Proc of the 6th Int Conf on Security of Information and Networks．New York：ACM，2013：233—24.

致谢
攻读学位期间发表的学术论文