学校代码 10530　　学　　号 200909021019

分 类 号 O241.82　　密　　级

湘潭大学

硕 士 学 位 论 文

基于条件随机场的多引擎云安全机制研究

学 位 申 请 人 杜 亚 娟

指 导 老 师 李明军 教授　程戈 副教授

学 院 名 称 数学与计算科学学院

学 科 专 业 计 算 数 学

研 究 方 向 信息处理及其应用软件

二〇一二年四月二十日

基于条件随机场的多引擎云安全机制研究

学 位 申 请 人 杜 亚 娟

导师姓名及职称 李明军 教授 程戈 副教授

学 院 名 称 数学与计算科学学院

学 科 专 业 计 算 数 学

研 究 方 向 信息处理及其应用软件

学位申请级别 理 学 硕 士

学位授予单位 湘 潭 大 学

论文提交日期 2012-4-20

Research of Multi-Engine Cloud Security Mechanism Based on Conditional Random Fields

Candidate Yajuan Du

Supervisor Professor Mingjun Li Ge Cheng

College School of Mathematics and Computational Science

Program Computational Mathematics

Specialization Information processing and its applications

Degree Master of Science

University Xiangtan University

Date April 20th, 2012

湘潭大学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湘潭大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

涉密论文按学校规定处理。

作者签名：　　日期：　　年　　月　　日

导师签名：　　日期：　　年　　月　　日

摘　　要

云安全是当前热门的研究方向，它使用多引擎检测，并将病毒数据库置于云端。它一方面克服了传统杀毒软件采用单一病毒检测策略，对相当数量的病毒无法检测的缺陷；另一方面，通过云端共享病毒库数据，一旦有新病毒产生，其信息即被整个网络识别。这使得病毒被发现和清除的时间更短，大大减少了病毒造成的损失。然而，现有的云安全架构仍存在一些不足。

首先，多引擎检测使用的热门技术主要有特征字节码技术和行为分析技术。它们一般基于病毒的字节码序列和系统调用序列的频次信息，使用数据挖掘或分类算法建立病毒模型，然后将该模型用于病毒识别。其不足之处在于，它们只使用了字节码序列和系统调用序列的频次信息，却忽略了各序列之间的次序关系。其次，对于恶意文件的海量序列信息，通常使用的方法是选择频次较高的部分序列。然而，一般病毒信息中病毒码的出现频次并不一定高，只选择频次高的序列必将忽视病毒的一些特征信息。再次，对于多引擎检测结果如何进行决策的问题，大多数采用投票分类方法，即将引擎数量较多的结果集作为最终的检测结果。该方法只考虑引擎数量，而忽略了各个引擎本身的检测准确度，导致误报误杀问题频繁产生。

本文主要研究基于条件随机场的云安全机制。首先，本文对当前已有的特征字节码和行为分析技术进行改进。在建模过程中，除了使用字节码和系统调用的频次信息之外，还考虑它们之间的次序信息，本研究中称该信息分别为次序字节码对和次序系统调用对。同时，使用信息增益理论对得到的频次信息进行筛选，选取信息增益较大的部分序列。然后，使用条件随机场理论，基于该静态和动态次序信息得到的病毒特征向量，为不同病毒种类建立条件随机场模型。将这些病毒模型用于可疑文件检测，得到对应的静态检测引擎和动态检测引擎的检测结果。最后，本文选取一些常用的病毒检测软件，将其和动静态检测引擎作为本研究的多引擎检测组。本文使用证据理论进行多检测结果融合。将各引擎的检测准确度作为各自的基本置信指派，使用证据合成法则得到最终的检测结果，降低了检测的误报率。

关键词：条件随机场；证据理论；静态检测；字节码；动态检测；系统调用；云安全；多引擎检测

Abstract

Nowadays,cloud security is a hot research direction where it makes use of multi-engines for malware detection and locates its virus database into the clouds. For one hand, it overcomes the disadvantages of traditional anti-virus software's single detection strategy which are at a loss what to do for quite a number of virus. For the other, it shares viruses' data through the clouds which makes the new virus be detected once it appears. This mechanism shorten the time of finding and clearing the new virus which reduces loss made by virus. However, there are still some shortcomings in current cloud security architecture.

Firstly, the common technologies for multi-engines are the N-grams feature extraction and behavioral analysis. Normally, they base on the frequencies of N-grams and system calls, make use of data mining methods and classification algorithm to model viruses' information and apply the model to detect viruses. The main shortage of them is that they just make use of the frequencies of N-grams sequences and system calls sequences, but ignore the sequential relationship among each sequence. Secondly, for the large sequence information of malware, the usually used method is to choose part of sequences with highest frequencies. However, normally the virus code in a virus file will not appear so frequently. Thus, this choosing method will ignore some features of virus. Thirdly, the decision based on multi-engines' detection results mainly uses voting classification method which takes the majority engines' results as the final results. Furthermore, this method just considers the differences of engines but ignores the detection accuracy of each engine, which will result in high false positives.

This paper mainly do research on the cloud security mechanism based on Conditional Random Fields. Firstly, this paper improves the current methods of N-grams feature extraction and behavioral analysis. In the modeling, besides of N-grams and system calls' own frequencies, it also uses frequencies of their orders which are respectively called pair of sequential N-grams and pair of sequential system calls in our research. Secondly, this paper puts the sifted information into the feature vector of such virus using the information gain theory. Thirdly, this paper models on these features using Conditional Random Fields and gets different virus models. If used for recognizing test files, the models will give different results which will be taken as the final results of static and dynamic engines. At last, this paper adds

some usual detecting softwares into the known two engines and puts them together as the engine group in our experiment. This paper uses D-S evidence theory in which it takes each engine's detection accuracy as their basic beliefs and uses D-S combination rule to get the final results, reducing the false positives.

Key words: Conditional Random Fields, D-S evidence theory, static detection, N-grams, dynamic detection, system calls, cloud security, multi-engine detection.

目 录

插图目录

表格目录

第一章　绪　论

§1.1　研究背景

随着信息技术的发展，现代社会越来越依赖于计算机系统。特别是近年来在互联网技术的推动，使得计算机平台的安全性显得愈发重要。然而，计算机安全问题一直随着计算机的发展而存在。据不完全统计显示，世界上大约有6亿以上的计算机用户。他们大部分通过局域网、无线局域网或万维网连接网络。基本上所有计算机用户都在遭受各种网络攻击的威胁，如蠕虫病毒、特洛伊木马及缓存溢出等。病毒的威胁日益猖獗，给互联网及计算机系统的使用造成了巨大的经济损失。根据文献[1]所示，网民受病毒感染入侵仅2010年上半年就达5.96亿人次，也就是说平均每天有331万网民遭受病毒的威胁。其中网络钓鱼软件尤为严重，造成的损失超过200亿元。计算机病毒通过稍微改变攻击策略，产生大量变种，并采取更加多样的传播途径。尤其人们逐步迈入网络时代，病毒的传播不再仅限于传统的局域网传播或者移动存储设备传播等方式，他们还可以通过电子邮件、网站以及一些即时的通讯工具传播。甚至说，计算机网络发展越快越多样，计算机病毒的传播就更容易，感染的范围也就越广。多样化的病毒逐渐成为计算机系统及网络的可用性、安全性和隐私保护性的最大威胁[3, 4]。

§1.1.1　计算机病毒防治

计算机网络的发展规模逐步增大，架构也日益复杂，使得病毒防范更加困难。很多网络攻击利用网络或者软件，甚至操作系统的漏洞和薄弱环节，采取各种方法窃取用户重要信息，进而导致整个网络系统的全面崩溃。《2009 年中国网民网络信息安全状况调查报告》[2]显示（如图1.1），71.9%的网民的浏览器配置被恶意软件修改，52%的网民曾遭受过互联网病毒攻击的威胁，50.1%的网民由于遭受恶意攻击，导致无法使用网络服务，45.0%的网民，他们的数据或者文件被病毒损坏或恶意删除，41.5%的网民，恶意攻击使其操作系统崩溃。据统计此类网络安全事件的诱因，网络下载或浏览占77.5%，电子邮件占3.5%，总计与网络相关的事件占安全事件的诱因超过80%。另外有10.1%的网民不清楚是何种原因感染病毒，目前，病毒隐蔽性越来越好，安全意识不太强的普通用户根本无法防范。

人们最常用的防护计算机和查杀病毒的方式是使用杀毒软件以及防火墙等这些传统的病毒防治技术。但这些传统技术存在一些弊端，如针对现在新出现的一些

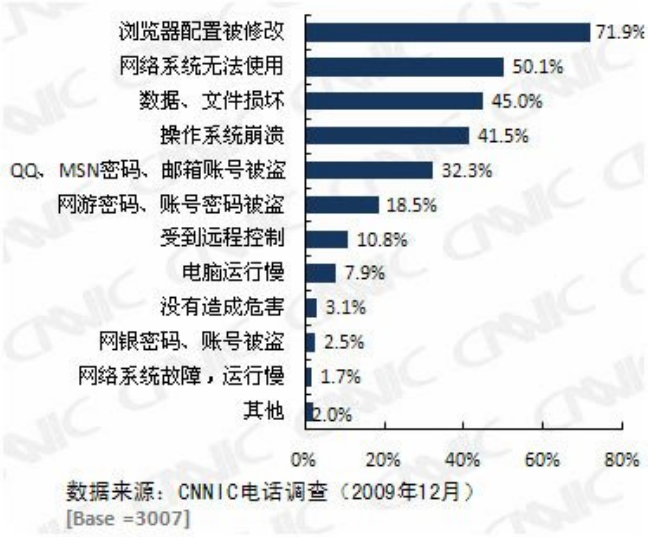

图 1.1 安全事件带来的危害

结构更加复杂的病毒，传统软件由于受到自身病毒检测算法局限性的影响，查杀效率大大降低。尤其对一些零生命周期的病毒，很难及时发现并清除。传统的杀毒软件主要通过识别病毒的特征码来查杀病毒，当杀毒软件发现可疑文件时，提取其特征码与病毒库中的病毒特征码进行比对，若发现病毒特征码，便判定该文件感染病毒。这种处理方法针对新变种病毒，必须适时更新病毒库，用户也必须经常下载更新病毒库。针对新出现的一些结构更加复杂的变种病毒，尤其一些零天生命周期病毒，传统软件受到自身病毒检测算法局限性的影响，根本无法知道其特征码，由此，查杀效率大大降低。当一种新病毒产生时，由于互联网的快速传播，甚至当杀毒软件尚未发现该病毒时，它已感染了大部分电脑及网络系统，此时传统的杀毒软件形同虚设。

此外，有些变种病毒甚至利用杀毒软件自身的漏洞和缺陷，侵入系统后直接攻击杀毒软件使其瘫痪，导致其对计算机的保护能力完全丧失。传统杀毒软件都有一套各自的查毒策略，即发现病毒特征码的算法，这种策略并不是每次都能正确查找病毒，很可能出现误报的现象，误杀用户的安全文件。同时，传统上使用的杀毒软件对终端资源占用量很大，难以满足移动互联网时代资源受限的移动端的需要。由此，基于云计算的安全架构成为学术研究和企业应用的趋势。本文基于云计算架构，提出一种云安全环境下的多引擎病毒检测方法。

§1.1.2 云安全概述

在基于云计算的安全领域，恶意代码检测行业是目前进展较快、影响较大的一个领域。云安全是恶意代码检测领域比较热门的研究方向。云安全来源于云计算技

术，是云计算在病毒查杀方面的应用。云安全将网络杀毒架构中需要在客户端处理的操作放到云端，一方面减轻了客户端的负载，另一方面云端的大数据量存储使病毒信息共享更即时。如常用的搜索引擎及邮件服务，就是这样的思想。本文将云计算架构用到网络安全中，客户端无需再保留病毒特征代码库，也不必适时下载更新病毒库。所有的信息都在云端存储，当全球的病毒库都集中到云端后，当新的变种病毒仅感染相当小的一部分文件时，其病毒特征信息便存储于云端数据库中，对其他文件进行病毒查杀时，便可快速发现该变种病毒。同时，大部分查杀处理操作都在云端完成，客户端无需按时杀毒和升级软件，大大减少了内存的占用。其主要的技术特点总结如下[9, 8, 10]：

(1) 提供超大规模的病毒查杀能力。例如，Google的超100万台的云计算服务器，它使用的200多个站点能够提供了强大的搜索引擎服务。将该模式运用到云安全中，对于复杂的病毒识别响应程序，云安全能够快速地提供计算和检测;

(2) 采用虚拟化技术。云安全使得用户可以使用任何客户端在任意位置获得病毒查杀服务，这种服务对用户来说，都是虚拟的，用户请求的资源来自于“云”;

(3) 通用扩展性。云安全的云端检测并不针对某种杀毒引擎，也不局限于某一种病毒查杀算法，可以采用多个杀毒引擎。在同一片云下，支持多种病毒查杀程序的开发及合并，且提供第三方开发环境;

(4) 病毒数据集中存储。在传统病毒查杀模式下，用户需要实时下载更新病毒库。在云安全环境下，所有的病毒信息能够集中在云端，使得病毒查杀更全面;

(5) 容错性。云端的多个杀毒引擎使得病毒攻击杀毒软件使其瘫痪的可能几乎为零。当某个引擎遭受攻击，停止工作时，其他引擎仍可以继续提供服务。

§1.2 研究现状

§1.2.1 计算机病毒检测的相关研究

由于计算机病毒日益猖獗，对人们对计算机的使用造成了极大地危害，一些盗号木马等病毒的出现更是对人们的信息安全造成了破坏性的威胁，病毒检测成为当今的热门研究方向。其中比较流行方法主要可以分为静态检测和动态检测两种。静态检测，即基于签名的病毒检测方法，它利用文件本身的字节码等特征，建立病毒特征库，在病毒尚未执行操作时，提取其特征码与病毒库比较，从而识别是否为病

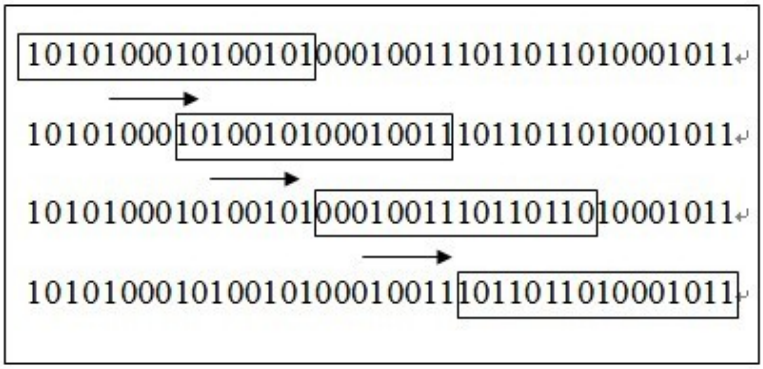

图 1.2 字节码滑动窗口（字节长度为2）

毒及其类型。该方法虽有较高的精确度，却对一些特征码稍加改动的变种病毒束手无策。这种方法最根本的问题在于它忽略了程序行为。动态监测方法基于病毒的行为特征，检测病毒的动态或静态行为。即使病毒出现新的变种，它们的攻击行为不会改变。然而，动态检测的不足在于，它只能当病毒对系统产生攻击后才被发现，有可能对系统造成危害。下面介绍几种当前热门的病毒检测技术。

§1.2.1.1 特征字节码技术

特征字节码是计算机病毒学研究领域基于序列的病毒码检测中最常用的基本特征[12, 13, 14, 15, 16]。特征字节码[17]是从能够表征某些代码实现特征的二进制可执行文件中提取的字节序列，它是一组连续的字节码。一般常用的字节码长度为1到3。特征字节码技术通常被用到很多技术中，它很容易理解，也便于实现。通过将字符串数据转化为字节码，能够非常高效方便地在向量空间比较数据流。也就是说，能够比较包含在某些数据集中的字节码分布特征，进而决定一些新的类似数据流的这些特征。

一般而言这些数据集称为“训练数据”，而对应的需要进行鉴定和判断的数据集称为“测试数据”。特征字节码技术正是通过对病毒文件或者正常文件的字节码分布进行建模，得到其特征，进而确定未知文件是否为可疑文件，甚至直接鉴定为病毒文件。在这个过程中，一般需要提取字节码的一些特征，然后采用决策树和机器学习的方法对其建模。除了对未知文件的静态代码进行特征提取外，对于其系统调用序列也同样可以采用该方法建模。

字节码分布通过在数据集中滑动固定大小的窗口，然后计算每个字节码在数据集序列中出现的次数。如下图1.2所示，给出长度为2个字节的特征字节码的滑动窗口，当滑动窗口向右移动一个字节，就产生一个新的字节码，每个字节码用“窗口”

标注。字节码的长度以及每次移动的长短取决于具体的应用。计算复杂度会随着字节码的长度成指数增长[12]。表1.1为字节码的频次统计示例。

表 1.1 字节码的频次统计

可执行文件	字节码特征				
	0180	523D	6060	E010	918A
win32.a	45	4	28	12	39
win32.b	23	34	12	59	27
win32.c	9	21	30	4	18
win32.d	11	76	41	33	52
win32.e	22	44	6	91	35
win32.f	5	7	32	30	4
win32.g	21	2	69	8	12

在提取文件的字节码频次特征后，为了更准确地表征不同文件，可以采取不同的特征提取方法：使用粗糙集理论降低特征的复杂度[12]；舍去某些字节码的频次，并计算不同长度的字节码频次，用以减少随长度增长引起的复杂计算[13]；计算字节码频次组成的向量的13种特征距离[14]；采用信息学理论得到字节码特征[15, 16]。特征代码法的常见实现步骤如图1.3的流程图所示：

- 首先，分析文件的字节码特征，主要做法为统计字节码的频次信息。
- 其次，对得到的字节码特征进行选择，使用信息学或者分类学方面的理论，过滤掉一部分无用信息，减少计算复杂度。
- 再次，使用数据挖掘或概率学等理论，对病毒或正常文件进行训练，得到某检测模型。
- 最后，由上步得到的检测模型，对测试文件进行病毒检测，给出检测结果。

由上述分析可见，现有的基于特征码的技术中大部分采用了相邻或不相邻的字节码信息，却忽略了字节的次序相关性，本文利用字节之间次序的相关信息，给出一种新的特征提取方法。

特征代码法有也有不足的地方，由于首先于病毒库数据，对于新的病毒，它必须实时更新版本，才能够进行检测识别。对新出现的病毒，尚未建立其代码特征库，难以对其建模，此时进行未知病毒检测将不可行。于是下面介绍一些基于病毒行为的分析技术。无论是未知或已知病毒，它们对计算机的攻击行为不会改变。

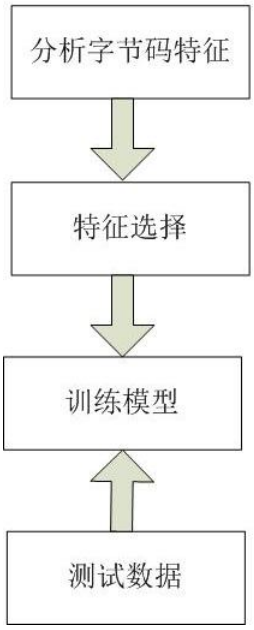

图 1.3 字节码分析技术的常见流程

§1.2.1.2 行为分析技术

病毒检测方法主要有静态和动态检测，如上所述的字节特征码技术可归类为静态检测，在文件未执行前，通过分析其静态代码鉴别是否为病毒。然而，它的缺陷在于其对字节匹配比较敏感，容易造成较高的误报率。而动态检测是基于病毒行为的检测技术，无论病毒的攻击性如何被掩饰，它们发动攻击时的行为不会改变，下面介绍行为分析技术方面的研究工作。

病毒的行为可以分为静态和动态行为，对应的分析技术可分为静态行为分析和动态行为分析[18]。当前，动态行为分析的相关研究较多[19, 20, 21, 22, 23] 21-24，也有静态行为的研究[18, 24, 25]。静态行为分析指在文件执行之前，对执行文件中的每一个可能的执行路径进行采用语义学分析技术进行分析，然后提取其执行特征，再使用分类算法鉴别其是否可能有恶意攻击行为。

动态行为分析技术一般采用的方法是首先对病毒程序进行分析，得到该程序对系统函数的调用序列，进而得到一组病毒特征向量，然后通过一系列向量比较算法，鉴定未知文件是否为病毒。病毒通常利用的攻击入口及其行为主要有下面几种[23]：

1. 未经验证的输入。如果一些网络请求数据，未经验证便进入网络系统。攻击者可能利用这些缺陷作为入口，通过网络应用对系统的后端部件进行攻击。一旦发现此类非法行为，即可将其作为病毒的行为特征。

2. 失效的访问控制。如果一些对已授权用户的限制不能被正确地强制执行，攻击者可能会利用这些缺陷存取其他用户的账户信息，查看关键文件或者使用未经授权的功能。正常文件通常不会有这样的操作，可以将其作为病毒特征。

3. 缓存溢出。在一些编程语言中，如果网络应用组件没有正确地对输入进行验证，可能导致程序崩溃。这些应用组件可能包括公共网关接口（CGI），函数库，驱动器和网络应用服务组件。

4. 不恰当的错误信息处理。指的是在常规操作下发生的错误情况没有没正确处理。如果攻击者能够引发网络应用不会进行处理的错误，它们可能会获得具体的系统信息，拒绝服务，导致安全机制丧失或者服务器崩溃。

5. 不安全的存储。网络应用通常使用加密函数来保护信息和证书。这些函数和代码一般很难正确编码，这便频繁地导致了系统的薄弱保护。

6. 拒绝服务。攻击者可能在某些其他合法用户不会使用的地方消耗网络服务资源。攻击者也可能锁定用户账户，甚至导致整个应用失效。

7. 对操作系统的基本配置，尤其是注册表进行修改，如增添开机启动项，注册新的服务等。

由上述的病毒行为，便可以以此为特征得到病毒的行为特征向量，进而判断文件是否为病毒。事实上，通常出现上述行为的病毒文件对系统的调用序列有一定规律的，本文对这些调用序列进行分析，通过系统调用函数的频次统计及其次序分布的规律统计，得到一种新的病毒行为检测方法。

§1.2.1.3 启发式扫描技术

启发式扫描技术通俗而言，即是将某种经验和知识，在病毒检测程序中体现，如一些熟练的程序员凭借对正常程序和病毒程序行为的经验，很容易觉察病毒程序。启发式指的是某种“运用某种方法和技术去判断某事物的能力”或者“自我发现的能力”。

病毒检测软件中若使用了启发式扫描技术，则它就相当于一个反编译器，只是以特定方式实现的而已，这种特定方式一般通过对有关指令序列的反编译进一步发现和理解其真正动机。启发式扫描技术在具体实现上有其独特的方式，它会将各种可以的程序代码按照可疑等级排序，并授予不同权值。如格式化磁盘、驻留内存以及搜索定位可执行程序等恶意操作。对于这个权值和要预先给定一个阀值，当检测

中权值和超过这个阀值时，便可启动病毒警报。具体实现中，一般会将多种程序并发的警报作为病毒存在的标识。

病毒的启发式分析技术一般分为两种：静态和动态。静态启发式技术一般使用非执行时间指示子，如结构化异常，程序分解和字节码[14]等技术。比较而言，动态启发式使用执行时间指示子，如商业沙箱应用或者反病毒产品的仿真性能。

虽然静态启发式在二十世纪九十年代取得了较大的成功，当前的研究和商业反病毒产品大大地推动了动态启发式的发展。反病毒公司使用静态和动态启发式启发算法于商业产品中。大多数静态启发式需要原始分解，这个是很难实现的。动态启发式避免了这个局限性，因为他不需要分解，只需要造限定的环境下观察程序执行过程一段具体的时间。观测程序行为要求提供所有程序的相关性。相比于静态启发式而言，这对于动态启发式要求更多。当必需的运行库缺失时，程序可能不会在测试环境下成功地执行。

动态启发式方法一般而言比静态启发式慢一些，因为他需要提前有一段观察时间和仿真。他的性能使得对于测试一个系统的十万单一程序，在操作上不可行。动态启发式也是不完备的，因为他不能保证观测时间恰好能观测到病毒行为。许多病毒样本要求触发机制来展示他们的恶意行为。例如，米开朗琪罗病毒只在3月6号，它产生的纪念日，执行其有效负载。

§1.2.2 云安全机制的相关研究

近年来，很多国内外知名的杀毒软件公司开始将云安全的架构作为其方向。如360杀毒，金山，瑞星及趋势科技等。当前主要有两种云安全实现模式，主要以趋势科技和瑞星的两种模式为代表，下面分别介绍其技术实现原理[5, 6, 7, 8]。

趋势科技的云安全产品架构主要建立于大量的服务器之上，把病毒特征码文件存储在云端庞大数据库中，在客户端只使用最低数量用于验证，使其尽量少地占用用户资源。该技术在趋势科技自身庞大的服务器群和并行处理能力的基础上，能够构建黑名单和白名单的服务器群，在恶意攻击到达用户电脑之前进行拦截。但是，趋势科技的云安全架构存在诸多的缺陷，它虽然能够对外来威胁进行判断和拦截，却无法对客户端已存在的未知威胁进行有效地响应和查杀。

瑞星的云安全架构采用的机制是称为“样本收集处理”。它通过大量的客户端检测网络中的异常行为，获取木马及恶意攻击等的最新信息，然后将其送至服务器端进行分析检测，分析检测得到的处理方案，再次对应地传送到每个客户端。由此可见，瑞星的云安全实现需要有大量客户端的支持，才能体现云的概念。瑞星的云安全计划核心之一是其自动在线检测模块[11]，该模块会在用户启动电脑后自动检

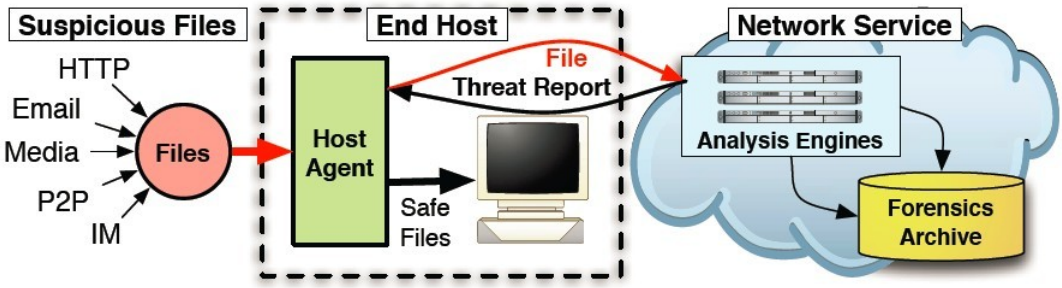

图 1.4 CloudAV架构

测，通过实时分析用户电脑上的文件更新和行为活动，将发现可疑文件或者可疑行为上传到瑞星自身提供的一种自动分析系统。这个过程耗时很短，完成处理后该分析系统会将对文件的分析结果重新发送到客户端，进而进行删除或者隔离病毒的操作。同时，将有毒或者恶意文件的信息存储在资料库里，这样该病毒或恶意软件的资料就被所有的用户电脑所共享。这样，每个的用户都能够提供一些有用的可疑文件信息，同时也能够分享利用其它用户提供的安全成果。可见，该系统参与者越多，整个网络就更安全。该机制也有不足之处，虽然它能检测客户端已存在的病毒行为，却无法对未知的外来入侵病毒进行拦截。

除以上所述的杀毒软件公司外，不少国内外研究学者也将云安全作为其研究方向。下面给出几种研究的具体情况。

Jon Oberheide等人[63]提出一种新的病毒检测架构CloudAV。该架构基于反病毒可作为网络服务以及多重防御的思想，突破了传统杀毒软件的局限性，解决了其诸多问题。该架构主要由终端主机和云端服务组成，如图1.4所示。终端主机主要由主机代理和终端电脑组成，它负责监测计算机上的各种活动，将可以文件信息发送至云端服务。云端服务主要由分析引擎和取证文档组成，它负责对终端主机的可疑文件进行检测，并发送检测报告至终端主机。它的反病毒可作为网络服务的思想，将传统杀毒软件客户端的大部分工作移至云端服务，用户只需按需购买杀毒服务就能够满足病毒查杀的需求，无需占用本机资源；它的多重防御思想，提供了多个检测引擎进行分析检测，一方面解决了传统软件容易受攻击的问题，另一方面使得检测准确度增加。然而，它没有解决的重要问题是，如何对多引擎的检测结果进行决策以及对海量的病毒数据如何优化采样。

Igor等人[29]描述了过去和当前基于网络云进行检测的安全技术，讨论了基于云的恶意扫描器如何与传统的扫描技术并存，以及这种云技术的优势所在。该研究给出了传统病毒检测软件对于病毒文件从发现到布置防御的步骤以及云安全系统

图 1.5 从病毒发现到部署防御的传统步骤

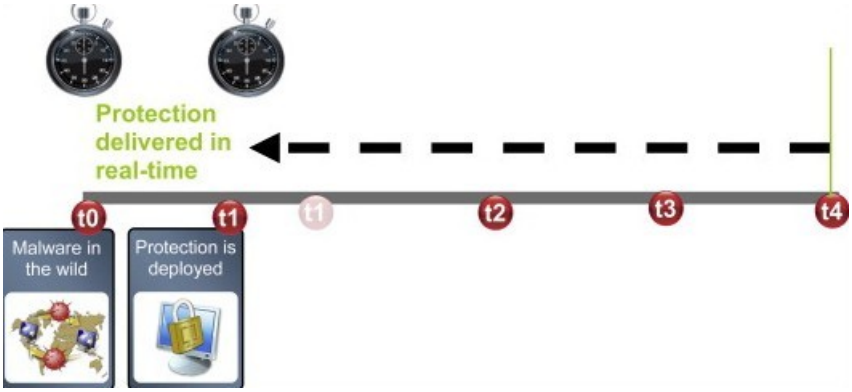

图 1.6 云安全系统减少了部署防御的时间

下部署防御的步骤，如图1.5和1.6所示。传统病毒检测软件一般需要四步来保护终端点不受恶意侵害。在每一步都有可能发生延迟，这样总延迟时间就使得病毒有机可乘，侵入用户电脑并造成损害。基于云的安全技术允许更快的反应，防御系统越早部署，病毒对用户造成的损害越小。该研究从执行速度、线下检测、带宽及安全性等七个方面阐述了云安全杀毒相比于传统杀毒软件的优势。然而，该研究依然没有提及多引擎决策和优化采样问题。

Schmidt等人[30]基于当前云计算的成功应用以及网格计算的发展，提出了虚拟技术的概念。虚拟化使得用户界面和用户控制之间更安全更灵活。他针对联合恶意软件检测和内核根目录工具箱防御，提出一种在虚拟的云计算环境下的检测方法。所有在一个虚拟化实例中运行的二进制文件都会被拦截，并被提交到一个或多个分析引擎中。一个完整的检测基于病毒的数字签名库，所有活跃的系统调用信息都可以用来检测未知的恶意软件。进一步而言，为了阻止入侵者对一个运行实例进行长期控制，提出了内核根目录工具箱方法，只有被授权的或者信任的内核模块才允许

在运行时被加载。这样加载未授权的模块就不可能了。该研究虽然没有明确提出云安全架构的概念，但是其使用的检测方法明显基于云安全的多引擎检测。但是该文也同样没有给出如何优化采样以及如何对多引擎检测结果决策。

§1.2.3 小结

本节主要介绍了计算机病毒检测方面以及云安全机制的相关研究。现有的研究方法大致可分为静态检测和动态监测，静态检测一般基于病毒文件自身的本身特征，在病毒程序未运行的时候，对其进行识别和检测。无论是病毒的特征字节码技术还是静态启发式扫描技术，都是基于文件这样的特征；动态检测是在文件运行后，根据其出现的一些异常特征，来检测文件鉴定其是否为病毒。病毒的行为分析技术和启发式扫描技术正属于这种类型。

静态检测能够检测大部分的病毒，但是对新出现的病毒类型或者是经过精细处理和掩饰，能够逃避这些静态检测算法的文件，常常束手无策。然而，无论病毒如何掩饰，它们的攻击行为往往不会有太多的变化。这就是动态检测技术要发挥作用的时候。动态检测能够检测病毒行为，却在施行过程中也有其局限性。这便需要一种机制，能够将动态和静态检测结果融合。本文提出的多引擎云安全机制正是实现了这一点，将静态和动态技术的检测结果，使用D-S证据理论融合，得到综合的检测结果。不仅减低了误报率，也提高了检测效率。在第四章具体介绍。

将上述检测方法融合在云安全机制中，它们构成了该机制的多引擎检测组。虽然很多公司和研究学者都提出了云安全架构，但是云安全的发展仍处于初级阶段，仅将客户端的病毒检测功能迁移到服务端，尚存在很多急需解决的问题。主要问题如下所述[6]：

- 如何在客户端和服务端对于入侵或是恶意数据进行优化采样。网络入侵种类和数量相当繁多，如果对于任何入侵都将其送至云端进行检测，必然会降低系统的效率，于是需要一种优化方法，筛除一些不必要的病毒信息。本文使用的信息增益理论即解决了信息筛选的问题。

- 如何通过服务端的多个监测引擎进行多决策。在云端，往往不同于传统病毒查杀软件，它一般有多个检测引擎同时对可以文件进行检测，这样就涉及到如何对多个检测结果进行多决策的问题，这也正是本文要研究的问题之一。

- 误报误杀问题。云安全的优势在于能够快速感应恶意攻击，并快速响应处理。然而，海量的可疑文件如果单纯靠自动检测，必然会出现误报误查的问题，有名的诺顿误杀事件就给了我们很大的警醒。应该采取相应措施降低误报率。本文

使用D-S证据理论进行多检测结果融合，一方面解决了多决策问题，另一方面通过多结果融合，提高了病毒的检测率，同时降低了误报率。我们将在第三章具体介绍。

§1.3 论文的主要研究工作

云计算技术的发展使得云安全成为当前网络安全领域的一个研究热点。云安全架构将大量病毒信息移至云端数据库中，降低了客户端的内存开销，同时实现了信息共享，使得新的病毒检测和查杀技术能够更快地得到广泛应用，也使得新病毒的危害得到更好地控制。本文正是基于这样的云安全架构，使用多引擎机制进行病毒检测。这些多引擎包含了当前流行的病毒查杀软件，同时将本文所提出的基于条件随机场理论的病毒检测技术运用其中，得到新的引擎检测结果，最后将其结果与已有的检测结果使用D-S证据理论融合，得到最终的病毒检测结果。本文的主要研究工作总结如下：

1. 基于云安全原理，构建了多引擎病毒检测机制。本文所构建的架构主要包括终端软件、云端服务器、云端管理器及病毒信息库。其中云端服务器使用Nimbus科学计算虚拟化工具进行模拟。

2. 提出一种基于病毒静态特征码检测和动态行为检测的改进方法。本文通过对病毒的静态字节码特征序列和动态行为序列（主要是系统调用序列）的分析，得到一系列病毒特征。使用条件随机场理论，在对其建模。将得到的病毒模型用于可疑文件的识别，识别结果作为云安全架构的静态引擎和动态引擎的检测结果。

3. 针对海量的病毒信息，使用信息增益理论对病毒码特征进行筛选，将信息增益较大的部分序列信息用于实验。

4. 对于当前流行的病毒查杀软件检测结果和本文技术实现得到的检测结果，采用D-S证据理论融合检测结果。该融合在提高检测率的同时，降低了系统对病毒的误报率。

§1.4 文章结构安排

基于以上研究内容，本文结构安排如下：

第一章首先介绍了计算机病毒防治的研究背景，简要介绍了云安全病毒查杀架构。再次，介绍计算机病毒检测和云安全机制的现有研究现状。最后，给出了本文的主要研究工作，介绍文章结构安排。

第二章介绍多引擎云安全机制使用的理论基础：条件随机场模型和证据理论。首先，给出条件随机场的应用背景及理论。在理论部分，本文给出了条件随机场的定义、条件概率以及训练和标记链式结构模型过程中用到的相关算法：迭代缩放算法和Viterbi算法。然后，介绍D-S证据理论，给出其基本概念以及证据合成法则，并通过一个简单的实例给出其在在实践中的具体应用。

第三章介绍了基于条件随机场的云安全机制的关键问题研究。本章介绍了本文的主要改进方法：改进的代码分析技术，海量病毒信息的筛选和优化采样，条件随机场特征的表示以及多引擎检测融合的机制研究。

第四章介绍基于条件随机场的多引擎云安全架构的实现，首先给出本文实验使用的系统架构，重点介绍了检测引擎和结果融合单元。其次给出介绍本研究实验使用的实验数据集和实验环境，并重点介绍本文的改进代码分析技术以及多引擎检测结果融合技术的实现。最后，分析本研究的实验结果。

第五章对本文的研究内容进行总结，提出了下一步研究方向和内容。

第二章　多引擎云安全机制的理论基础

§2.1　条件随机场

条件随机场（Conditional Random Fields,CRFs）是一种用于标记和分割关系数据的概率框架，它是用于结构数据分类的无向图模型[36]。通常使用训练数据得到模型，然后给定输入序列，使输出序列的条件概率最大化。CRFs适用于观测结果的标签分布建模，被称为有差别模型。CRFs的优势在于它能够灵活的合并任意重叠或聚合的观测特征，能够很好地提取序列的外部特征。CRFs吸引了不同领域的研究学者，并在很多项目中得到了成功运用，它的应用成果扩展到中文实体命名识别，文本分割，语音信息处理以及文献信息提取等不同的领域[38, 39, 40, 41, 42, 43, 44, 45, 46]。

§2.1.1　条件随机场理论

有关序列标记的算法通常用来表示联合概率分布$p(\boldsymbol{y}, \boldsymbol{x})$，其中$\boldsymbol{y}$表示要建模的实体，$\boldsymbol{x}$表示观测值。对联合概率分布建模通常很难实现，它一般要求得到分布函数$p(\boldsymbol{x})$。而一般情况下$\boldsymbol{x}$和$\boldsymbol{y}$之间有相当复杂的依赖关系，若是忽略这些关系，实验结果会大大失真，考虑这些依赖，得到$p(\boldsymbol{x})$是很困难的。解决该问题的方法之一是直接对条件概率$p(\boldsymbol{x}|\boldsymbol{y})$建模，将它用于数据分类已经足够。条件随机场（CRFs）[36]便是一种简单的条件分布。这样不必精确地表示变量的依赖关系，便可充分使用输入数据的大量特征。CRFs通过以观察数据为条件的无向图对变量建模。

§2.1.1.1　CRF的定义

假设，$\boldsymbol{x}$是用于描述观测数据的输入向量，$\boldsymbol{y}$是对应标签序列的随机变量，变量$\boldsymbol{x}$和$\boldsymbol{y}$是联合分布的。但在一个无差别框架下，通过成对的观测和标签序列建立条件模型$p(\boldsymbol{y}|\boldsymbol{x})$，而不必明确得到边缘分布$p(\boldsymbol{x})$。CRF的定义如下[36]：

定义 2.1.1 设$G = (V, E)$是一个图，其中V是顶点集，E是边的集合。$\boldsymbol{y}$为顶点的索引。如果随机变量$\boldsymbol{y}$在$\boldsymbol{x}$的条件下满足马尔科夫性：

$$p(y_i|\boldsymbol{x}, y_{v-i}) = p(y_i|\boldsymbol{x}, y_{N_i}) \tag{2.1}$$

则每一对$(\boldsymbol{x}, \boldsymbol{y})$称为一个条件随机场*(CRF)*。其中$v - i$是除顶点$i$之外的$G$中所有顶点，$N_i$是$G$中顶点$i$的相邻顶点集，$y_\Omega$表示集合$\Omega$中的顶点标签。

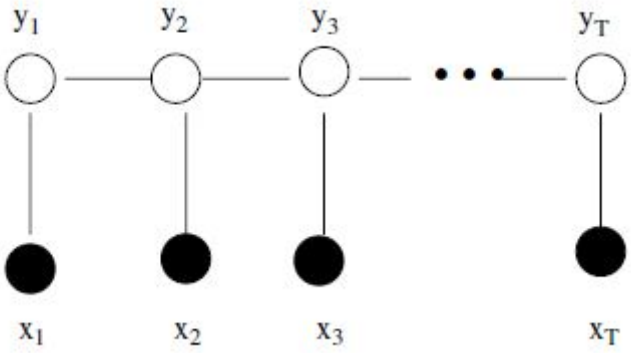

图 2.1 链式结构的CRF模型图

§2.1.1.2 条件概率的表示

由Hammersley - Clifford定理[37]可知:

定理 2.1.1 给定观测值$\boldsymbol{x}$，标签值$\boldsymbol{y}$的条件概率$p(\boldsymbol{y}|\boldsymbol{x})$ 的表达形式具有马尔科夫随机场的形式:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{\prod_{c\in C}\phi(\boldsymbol{y}_c,\boldsymbol{x})}{Z(\boldsymbol{x})} = \frac{\prod_{c\in C} exp\Big(\sum_k \lambda_k f_k(c,\boldsymbol{y}_c,\boldsymbol{x})\Big)}{Z(\boldsymbol{x})} = \frac{exp\Big(\sum_{c\in C}\sum_k \lambda_k f_k(c,\boldsymbol{y}_c,\boldsymbol{x})\Big)}{Z(\boldsymbol{x})} \tag{2.2}$$

其中c表示全连通子图，C是其集合。$\phi(\boldsymbol{y}_c,\boldsymbol{x})$称为$c$的潜在函数（势函数），它是$c$上的特征和的指数函数：$\phi(\boldsymbol{y}_c,\boldsymbol{x}) = exp\Big(\sum_k \lambda_k f_k(c,\boldsymbol{y}_c,\boldsymbol{x})\Big)$。$Z(\boldsymbol{x})$是归一化因子，它的表达式为$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} exp\Big(\sum_{c\in C}\phi(\boldsymbol{y}'_c,\boldsymbol{x})\Big)$，$\boldsymbol{y}'$因标签的不同形式而不同。

最简单也是最常用的CRF模型为链式结构，如图2.1所示，它符合有限自动机且适用于标签标记。对一组标签序列$\boldsymbol{y} = y_1 \dots y_T$，给定一系列输入序列$\boldsymbol{x} = x_1 \dots x_T$，参数为$\Lambda = (\lambda_1, \lambda_2 \dots \mu_1, \mu_2, \dots)$的链式CRF的条件概率可以表示为:

$$p(\boldsymbol{y}|\boldsymbol{x},\Lambda) = \frac{1}{Z(\boldsymbol{x})} exp\left(\sum_{t=1}^{T}\left(\sum_k \lambda_k f_k(y_{t-1},y_t,x,t) + \sum_{kk}\mu_{kk} g_{kk}(y_t,x,t)\right)\right) \tag{2.3}$$

其中$f_k(y_{t-1},y_t,x,t)$是在位置t和$t-1$处标签的过渡特征，$g_{kk}(y_t,x,t)$是整个观测序列$\boldsymbol{x}$和标签y_t的状态特征。它们一般为二值函数。例如:

$$f_k(y_{t-1},y_t,x,t) = \begin{cases} 1 & \text{if rainy today} \\ 0 & \text{otherwise} \end{cases}$$

$$g_{kk}(y_t, x, t) = \begin{cases} 1 & \text{if sunny today} \\ 0 & \text{otherwise} \end{cases}$$

T为序列的长度，λ_k和$\mu_k k$为模型的参数，在学习的过程中调整。

$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} exp\left(\sum_{t=1}^{T} \left(\sum_k \lambda_k f_k(y'_{t-1}, y'_t, x, t) + \sum_{kk} \mu_{kk} g_{kk}(y'_t, x, t) \right) \right)$为归一化因子。为简便起见，$g_{kk}(y_t, x, t)$也可以写成$f_k(y_{t-1}, y_t, x, t)$的形式，可以以相同方式处理这两种特征。

于是，给定特征集合S，CRF为：

$$P(\boldsymbol{y}|\boldsymbol{x}, \Lambda) = \frac{1}{Z_S(\boldsymbol{x})} exp\left(\sum_{t=1}^{T} \left(\sum_{f_k \in S} \lambda_k f_k(y_{t-1}, y_t, x, t) \right) \right) \tag{2.4}$$

其中，$Z_S(\boldsymbol{x}) = \sum_{\boldsymbol{y}'} exp\left(\sum_{t=1}^{T} \left(\sum_{f_k \in S} \lambda_k f_k(y'_{t-1}, y'_t, x, t) \right) \right)$

给定上述公式(2.4)描述的CRFs模型，在一定的输入序列下，需要找到最大概率的标签序列，此即：

$$y^* = argmax_y P(\boldsymbol{y}|\boldsymbol{x}, \Lambda) \tag{2.5}$$

这个最大概率可以使用动态规划中的Viterbi算法得到。通常使用最大对数似然估计来得到最大概率标签序列，在下面两小节中将有介绍。

§2.1.2 训练链式CRFs

对上式(2.5)中参数Λ的估计通常使用一些样本数据的观测量或者函数来实现。一般使用的估计量是最大似然估计。

对于一组相互独立的训练样本集$(\boldsymbol{x}^k, \boldsymbol{y}^k)$，其中$k = 1, 2, \ldots N$，对某个CRF模型$P(\boldsymbol{y}|\boldsymbol{x}, \Lambda)$ 训练数据的最大似然函数为：

$$M(\lambda) = \Pi_{\boldsymbol{x},\boldsymbol{y}} \log P(\boldsymbol{y}|\boldsymbol{x}, \Lambda) \tag{2.6}$$

对上式(2.6)两边取对数得到训练数据的条件对数似然函数。由对数函数的单调性，使下面式子取最大值Λ值即为式(2.6)中需要求的Λ。正则条件常被加入到条件对数似然函数中来避免过学习问题，下面使用高斯先验分布作为正则条件。

于是目标是使得下面的函数取得最大值：

$$\begin{aligned} L(\lambda) &= \sum_k \log p_\lambda(\boldsymbol{y}^k|\boldsymbol{x}^k) - \sum_j \frac{\lambda_j^2}{2\sigma^2} \\ &= \sum_k \Big[\sum_{i=1}^{T} \sum_j \lambda_j f_j(\boldsymbol{y}^k{}_{i-1}, \boldsymbol{y}^k{}_i, \boldsymbol{x}^k, i) - \log Z(\boldsymbol{x}^k) \Big] - \sum_j \frac{\lambda_j^2}{2\sigma^2} \end{aligned} \tag{2.7}$$

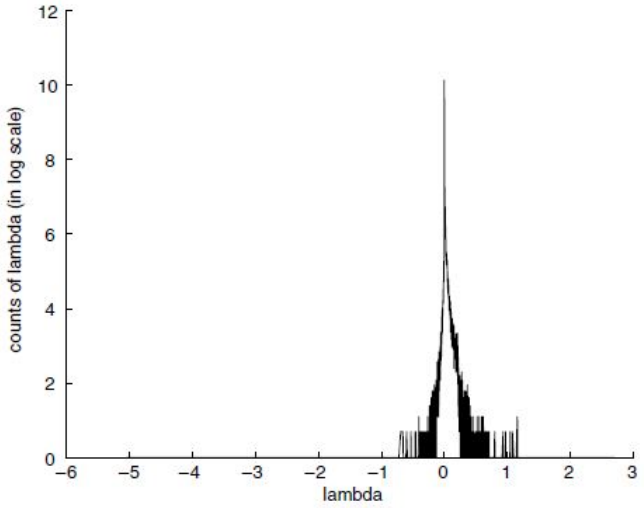

图 2.2 Λ的经验分布

其中，k表示训练数据$(\boldsymbol{x}^k, \boldsymbol{y}^k)$的索引序号，$\sigma^2$表示球面高斯先验分布的方差。$\lambda_j$为特征$f_j$的系数，需要在训练过程中得到优化。

上述$Z(\boldsymbol{x}^k) = \sum_{\boldsymbol{y}'} exp\left(\sum_{t=1}^{T} \sum_j \lambda_j f_j(y'_{t-1}, y'_t, x^k, t) \right)$。

上式(2.7)对变量λ_j求偏导数，得到：

$$\begin{aligned} \frac{\partial L(\lambda)}{\partial \lambda_j} &= \sum_k F_j(\boldsymbol{y}^k, \boldsymbol{x}^k) - \sum_k \sum_{\boldsymbol{y}} P_\lambda(\boldsymbol{y}|\boldsymbol{x}^k) F_j(\boldsymbol{y}, \boldsymbol{x}^k) - \frac{\lambda_j}{\sigma^2} \\ &= E_{p(\boldsymbol{y},\boldsymbol{x})}[F_j(\boldsymbol{y}, \boldsymbol{x})] - E_{p(\boldsymbol{y},\boldsymbol{x},\lambda)}[F_j(\boldsymbol{y}, \boldsymbol{x}^k)] - \frac{\lambda_j}{\sigma^2} \end{aligned} \tag{2.8}$$

其中，$F_j(\boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1}^{T} f_j(\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{x}, i)$ 是给定$\boldsymbol{x}, \boldsymbol{y}$之后，第$j$个特征的“计数”。$E_{p(\boldsymbol{y},\boldsymbol{x})}[F_j(\boldsymbol{y}, \boldsymbol{x})]$ 表示特征$f_j(\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{x}, i)$对训练数据经验分布的期望，Λ的经验分布如图所示[43]。

$E_{p(\boldsymbol{y},\boldsymbol{x},\lambda)}[F_j(\boldsymbol{y}, \boldsymbol{x}(k))]$表示在模型分布下特征$j$的期望。

变量λ_j的最优值往往不能通过令式(2.8)等于零求解得到。所以，大量迭代算法被用于求CRFs模型中特征系数最优值。Lafferty[36]使用迭代缩放算法。该方法简单可行，且能保证收敛性，然而对一些大型项目计算速度太慢。牛顿方法，例如L-BFGS算法，目前被认为是一种更高效的算法[35]。此外，Vishwanathan[34]使用随机梯度最大化算法来训练CRFs模型。下面给出迭代缩放算法。

迭代缩放算法

在上述问题中，需要找到使得公式(2.7)取最大值的Λ。迭代缩放算法趋向于每次迭代对Λ施加一个增量$\delta = (\delta_1, \delta_2, \dots \delta_T)$，使得$L(\Lambda + \delta) - L(\Lambda)$最大。由对任意的$x$，满足$1 - x \leq \log x$，可知：

$$
\begin{aligned}
&L(\Lambda+\delta) - L(\Lambda) \\
&= \sum_k \Big[\sum_{i=1}^{T} \sum_j \delta_j f_j(\boldsymbol{y}^k{}_{i-1}, \boldsymbol{y}^k{}_i, \boldsymbol{x}^k, i) - \log \frac{Z_{\Lambda'}(\boldsymbol{x}^k)}{Z_{\Lambda}(\boldsymbol{x}^k)} \Big] - \sum_j \frac{\delta_j^2 + 2\lambda_j \delta_j}{2\sigma^2} \\
&\geq \sum_k \Big[\sum_{i=1}^{T} \sum_j \delta_j f_j(\boldsymbol{y}^k{}_{i-1}, \boldsymbol{y}^k{}_i, \boldsymbol{x}^k, i) + N - \frac{Z_{\Lambda'}(\boldsymbol{x}^k)}{Z_{\Lambda}(\boldsymbol{x}^k)} \Big] - \sum_j \frac{\delta_j^2 + 2\lambda_j \delta_j}{2\sigma^2} \\
&= \sum_k \Big[\sum_{i=1}^{T} \sum_j \delta_j f_j(\boldsymbol{y}^k{}_{i-1}, \boldsymbol{y}^k{}_i, \boldsymbol{x}^k, i) + N - \\
&\sum_y p_\Lambda(y|x) exp(\sum_{i=1}^{T} \sum_j \delta_j f_j(\boldsymbol{y}^k{}_{i-1}, \boldsymbol{y}^k{}_i, \boldsymbol{x}^k, i)) \Big] - \sum_j \frac{\delta_j^2 + 2\lambda_j \delta_j}{2\sigma^2}
\end{aligned}
\tag{2.9}
$$

其中，$\Lambda' = \Lambda + \delta$。

令$f_j(x,y) = \sum_{i=1}^{T} \delta_j f_j(\boldsymbol{y}^k{}_{i-1}, \boldsymbol{y}^k{}_i, \boldsymbol{x}^k, i)$，$f(x,y) = \sum_j f_j(x,y)$，由Jensen不等式得：

$$
\begin{aligned}
exp \sum_j \delta_j f_j(x,y) &= exp\Big\{ f(x,y) \sum_j \delta_j \frac{f_j(x,y)}{f(x,y)} \Big\} \\
&\leq \Big\{ \sum_j \frac{f_j(x,y)}{f(x,y)} e^{\delta_i} \Big\}^{f(x,y)} \\
&\leq \sum_j \frac{f_j(x,y)}{f(x,y)} e^{\delta_i f(x,y)}
\end{aligned}
\tag{2.10}
$$

由公式(2.9)和公式(2.10)，可以得到$L(\Lambda + \delta) - L(\Lambda)$的一个下界，记作$B_\Lambda(\delta)$。

$$
\begin{aligned}
B_\Lambda(\delta) = &\sum_{x,y} \sum_j \delta_j f_j(x,y) + N - \\
&\sum_{x,y} p_\Lambda(y|x) \sum_j \frac{f_j(x,y)}{f(x,y)} e^{\delta_i f(x,y)}
\end{aligned}
\tag{2.11}
$$

为使上式(2.11)取最大值，求解$\frac{\partial B_\lambda}{\partial \delta_j} = 0$，得到：

$$
\sum_{x,y} f_j(x,y) \Big\{ 1 - p_\Lambda(y|x) e^{\delta_i f(x,y)} \Big\} = 0 \tag{2.12}
$$

由上式(2.12)可得到δ的值，从而使迭代继续进行直至收敛到某个Λ。整个算法描述为：

算法 2.1.1 (迭代缩放算法) 给定模型(2.7)，求使该函数取最大值的Λ的计算过程如下：

1. 给Λ赋初值Λ_0

2. 重复下列循环至收敛：

   a. 求解$\frac{\partial B_\lambda}{\partial \delta_j}=0$，即式(2.12)，得到$\delta_j$的值。

   b. 令$\lambda_j=\lambda_j+\delta_j$

   注意到以下事实：

(1) 由式(2.12)，δ_j能够被独立计算出来。

(1) 迭代缩放算法能够达到其极值。

(3) 为达到最优的Λ，可以使用登山算法。

§2.1.3 标记链式CRFs

标记链式CRFs即对于给定观测值序列，找到最大概率的标签序列，也就是$y^*=argmax_y P(\boldsymbol{y}|\boldsymbol{x},\Lambda)$。Viterbi算法能够高效地解决这个问题。

Viterbi算法是用于找到隐藏状态的最大可能标记序列的动态规划算法。它通过迭代计算$\alpha_{t+1}(y) = max_{y'}\alpha_t(y')\cdot exp(\sum_j \lambda_j f_j(y',y,x,t))$来预测位置$t$的标签使得$y_t^* = argmax_{y'}\alpha_t(y')\cdot exp\Big(\sum_j \lambda_j f_j(y',y,x,t)\Big)$，计算$\alpha_{t+1}(y)$的时间与特征$f_j$的数量成正比。所以特征越少，预测标签的时间越短。下面给出Viterbi算法。

算法 2.1.2 (Viterbi算法) 设特征个数为N。

1. 给定$\alpha_1(y')$初值α_0。

2. 重复下列循环至所有特征：

$$\alpha_{t+1}(y)=max_{y'}\alpha_t(y')\cdot exp(\sum_j \lambda_j f_j(y',y,x,t)) \tag{2.13}$$

   并对于每个$t\in[0,N]$：

$$y_{t+1}^*=\alpha_{t+1}(y) \tag{2.14}$$

3. 所求标记序列即为：

$$y^*=(y_1^*,y_2^*,\dots y_N^*) \tag{2.15}$$

§2.2 D-S证据理论

在信息融合系统中，实践遇到的情况经常不像信息理论描述得那么理想化。各部分提供的信息可能是不完整不精确的，它们在某种程度上有相互的叠加和依赖，甚至彼此之间可能出现矛盾的情况。此时便需要一些不确定推理方法，目前，这些方法有：D-S证据理论、贝叶斯推理、粗糙集理论等。在本文中，使用D-S证据理论进行多引擎检测结果融合。D-S证据理论由Dempster[47]于1968年提出的上下概率体系发展而来，他的学生Shafer于1976年将置信函数的全面解释加入其理论中使其更加系统化、理论化。D-S证据理论近些年来在多数据决策领域[49, 50, 51]，图像信息领域[52, 53, 54]以及计算机科学领域[55, 56, 57]得到了广泛的应用。下面介绍D-S证据理论的基本概念。

§2.2.1 基本概念

对于一个决策问题，将人们能够认识到的所有可能结果集用$\Theta = x_1, x_2 \dots x_n$来表示，它称为识别框架（*Frame of Discernment*），其中的元素被称为问题的假设。它的选择依赖于现有的知识水平、已经知道的及想要知道的。识别框架Θ的子集称为一个命题（*Proposition*），则2^Θ 表示所有命题的集合，可以在该集合上定义并、交、补等运算。识别框架是证据理论的基础，与该理论相关的所有函数和概念以及证据合成法则都是基于识别框架的。

定义 2.2.1 设识别框架Θ，其上的基本置信指派（*Basic Belief Assignment, BBA*）定义为$m : 2^\Theta \to [0, 1]$，它满足下列条件：

$$m(\varnothing) = 0 \qquad and \qquad \sum_{A \subseteq 2^\Theta} m(A) = 1 \tag{2.16}$$

对于任意的$A \subseteq 2^\Theta, m(A)$也称为基本概率指派（BPA），它表示指派给命题A的概率大小，即支持命题A发生的程度。$m(A)$也称为质量（mass）函数。

定义 2.2.2 设识别框架Θ，由基本置信指派导出的置信函数（*Belief Function, BF*）定义为$Bel : 2^\Theta \to [0, 1]$，且满足：

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{2.17}$$

由上述两个定义可知，证据的基本置信指派为命题A的发生程度，不包含其子集的置信度；而证据的置信函数则为其所有子集的置信度之和。

定理 2.2.1 设识别框架Θ，集合函数$Bel: 2^{\Theta} \to [0,1]$为置信函数当且仅当它满足：

1. $Bel(\varnothing) = 0$
2. $Bel(\Theta) = 1$
3. 对任意的自然数n，$A_1, A_2, \ldots, A_n \subseteq 2^{\Theta}$

$$Bel(A_1 \cup A_2 \cup \ldots \cup A_n) \geq \sum_{i} Bel(A_i) - \sum_{i>j} Bel(A_i \cap A_j) + \ldots + (-1)^n Bel(A_1 \cap A_2 \cap \ldots \cap A_n) \tag{2.18}$$

在经典概率论里，概率满足可加性：

$$\forall A, B \subseteq \Theta, \qquad P(A \cap B) = P(A) + P(B)$$

由概率可加性知，若一个命题为真的概率为s，则必须以概率为1-s去相信该命题的否命题。在实际情况中，这样的条件是不成立的。如命题"明天会下雨"的概率为0.1，就得相信明天不下雨的程度为0.8。事实上，受限于天气预报技术或者过于相信天气规律，概率可加是不可信的。

于是，Shafer舍弃了概率可加性原则，取而代之得到半可加性，其定义参见上述定理第三条中的公式(2.18)。

定义 2.2.3 设识别框架Θ，由基本置信指派导出的似真函数（*Plausibility Function*）定义为$Pl: 2^{\Theta} \to [0,1]$，且满足：

$$Pl(A) = \sum_{B \cap A \neq \varnothing} m(B) \tag{2.19}$$

似真函数为与A的交集不为空的集合的基本置信指派之和，也就是不反对A的命题发生的程度。

上述的概念很明显具有一定的相关性，能够相互推导。例如：

$$Bel(A) = 1 - Pl(\neg A), \qquad Pl(A) = 1 - Bel(\neg A) \tag{2.20}$$

这里$\neg A$表示A的否命题，通常$Bel(\neg A)$表示对A的怀疑程度。其他类似关系如：

$$Bel(A) + Bel(\neg A) \leq 1, \qquad Pl(A) + Pl(\neg A) \leq 1 \tag{2.21}$$

上述两个不等式也同样说明了D-S证据理论与经典贝叶斯方法的主要不同之处。

定义 2.2.4 设识别框架Θ，由基本置信指派导出的公共函数（*Commonality Function*）定义为$Q: 2^{\Theta} \to [0,1]$，且满足：

$$Q(A) = \sum_{A \subseteq B \subseteq \Theta} m(B) \tag{2.22}$$

$Q(A)$没有明确的意义，有定义看出，公共函数反映了包含A的所有集合的基本置信指派之和。

定义 2.2.5 设识别框架Θ，A为Θ的子集

1. $(m(A), A)$称为证据体，若干个证据体组成了证据；
2. 若$m(A) > 0$，则称A为证据的焦点元素（*Focal Element*）；
3. 全体焦元的集合称为证据的核（*Core*）；
4. 若置信函数只有唯一的焦元Θ，则称为空置信函数（*Vacuous Belief Function*）；
5. 置信函数被称为贝叶斯置信函数（*Bayesian Belief Function*），当所有焦元都是单假设集时。

一般而言，人们对一个决策问题进行证据处理的过程可简化为如下数学模型：

- 首先，确定识别框架，对命题问题的研究转化为数学上对集合的研究；
- 其次，确定所有可能命题即识别框架的所有子集的置信指派、置信函数和似真函数，一般需要证据处理人员对现有证据进行分析，得到证据每个子集的支持程度；
- 再次，利用下节即将介绍的合成法则得到证据融合下的置信指派、置信函数和似真函数；
- 最后，根据决策规则，选择由证据合成法则计算得到的证据融合结果最大的假设。

§2.2.2 Dempster合成法则及应用

在现实应用中，往往有两个或两个以上的证据源，需要一种法则来合成这些证据源，以得到整体的置信函数，这个法则便是Dempster 合成法则。一般而言，两个证据的合成比更多证据的合成要简单，计算量也不会太大。通常往往将多个证据源的情况转化为两两合成的情况。在本节中，只介绍两个证据源合成的法则。

定理 2.2.2 设Bel_1, Bel_2是同一识别框架Θ上的两个置信函数，m_1, m_2为其对应的基本置信指派，其焦元分别为$A_1, \dots A_k$和$B_1, \dots B_l$ 如果两个置信函数之间相互独立，且

$$\sum_{A_i \cap B_j = A} m_1(A_i) m_2(B_j) < 1$$

那么，函数$m : 2^{\Theta} \to [0,1]$对所有的非空集合$A \subseteq \Theta$满足$m(\varnothing) = 0$ 且

$$\begin{cases} m(A) = \frac{\sum_{A_i \cap B_j = A} m_1(A_i) m_2(B_j)}{1 - \sum_{A_i \cap B_j \neq \varnothing} m_1(A_i) m_2(B_j)} = \frac{1}{M} \sum_{A_i \cap B_j = A} m_1(A_i) m_2(B_j) & A \neq \varnothing \\ 0 & A = \varnothing \end{cases} \tag{2.23}$$

且此函数为基本置信指派函数，其中

$$M = 1 - \sum_{A_i \cap B_j \neq \varnothing} m_1(A_i) m_2(B_j) > 0$$

记$m = m_1 \oplus m_2$。

证 明 $m(\varnothing) = 0$显然成立，现证$m(A) = 1$。

由于

$$\begin{aligned} \sum_{A \subseteq \Theta} m(A) &= m(\varnothing) + \sum_{A \subseteq \Theta, A \neq \varnothing} m(A) \\ &= \sum_{A \subseteq \Theta, A \neq \varnothing} \frac{1}{N} \sum_{A_i \cap B_j = A} m_1(A_i) m_2(B_j) \\ &= \frac{1}{N} \sum_{A_i \cap B_j \neq \varnothing} m_1(A_i) m_2(B_j) = \frac{1}{N} \cdot N = 1 \end{aligned} \tag{2.24}$$

∎

应用实例

了解了证据理论的基本概念及其证据合成法则，为了便于理解，来看一个应用实例。在下一章中，将用此理论为本文的多引擎检测结果建模，得到最终的实验结果。

假设要通过今天的天气状况来预测城市A明天中午的天气。设有三种天气状态：雨天、阴天和晴天，简记为R、C和S。则该问题的识别框架为Θ=（R,C,S）。现在假设已经搜集了两条证据，如下：

- 今天的温度在零度以下；
- 今天的大气压下降，也就是说可能有降雨来临

这两条证据的置信指派用m_1, m_2来表示，它的分布如表2.1所示：

表 2.1 两条独立证据的置信指派分布

	∅	(R)	(C)	(S)	(R,C)	(R,S)	(C,S)	(R,C,S)
m_1	0	0.2	0.1	0.1	0.2	0.1	0.1	0.2
m_2	0	0.1	0.2	0.1	0.3	0.1	0.1	0.1

由定义知，空命题不含有任何置信指派。m_1和m_2将它的信质分布到Θ的各个真子集上。使用m表示m_1, m_2 组成的联合证据的基本置信指派。假设上述两条证据相互独立，由Dempster合成法则（公式(2.23)）得到m的置信指派分布，结果见表2.2。

表 2.2 合成证据的置信指派分布

	∅	(R)	(C)	(S)	(R,C)	(R,S)	(C,S)	(R,C,S)
m	0	0.282	0.282	0.128	0.180	0.051	0.051	0.026

计算得到的值在元素（R）、（C）及（S）上分配了更大的概率。其他的两个或两个以上的状态组合的概率降低了，而且比任一个证据指派的概率值都小。

§2.3 小结

本章主要主要介绍了本研究使用的理论基础，首先介绍了条件随机场的相关应用，基本概念以及与它的实现相关的一些算法；其次，给出D-S证据理论基本概念及其中最重要的Dempster证据合成法则，并通过简单的实例给出其在实际中的应用。下面分别给出这两种理论的总结。条件随机场模型为序列标记任务另辟蹊径，使其不再局限于求解观测值和标签序列的联合概率分布。在实际应用中，一般使用链式结构的CRFs。它的实现主要可以分为以下几个步骤：

- 首先根据实际应用，得到能够体现识别任务之间不同之处的一系列特征；
- 然后在这些特征的基础上建立条件随机场模型，其中最重要的工作是训练CRFs的特征参数。通常对于小型项目，规模不大的情况下，使用迭代缩放算法。它具有较高的准确率，也比较容易理解。在一些大型项目中，使用L-BFGS等算法训练模型；
- 根据得到的训练模型，需要得到在该模型下使得条件概率取最大值的标签序列，该过程一般使用Viterbi算法实现。
- 将得到的标签序列返回模型中，确定需要得到的建模结果。

在证据理论中有几个基本概念。识别框架由所有可能命题组成，它是整个证据理论的基础。置信指派非零的元素构成了证据的焦元，基于置信指派导出的证据描述函数有置信函数、似真函数和公共函数。置信函数表示分配到某个焦元上的所有置信指派，它为其所有子集的置信指派之和；似真函数表示不反对某焦元发生的程度；公共函数反映了包含焦元的所有命题的置信度之和。

D-S证据理论相比经典贝叶斯理论而言，具有更普遍的应用，也更加能够反映实际情况，得到想要的预测结果。由上述分析和实例可以看到，当置信函数和似真函数相等时，它满足贝叶斯经典理论，这是D-S证据理论的一个特例。正如先前提到的，两者的区别在于，证据理论给出一种不确定情况的推理方法。

第三章　基于条件随机场的云安全机制的关键问题研究

本节将要研究条件随机场中一个非常重要的问题：针对特定任务，如何为模型选择合适的特征。下面先介绍我们的特征信息来源，即代码的静态和动态分析技术。

在上述§1.2.1.1和§1.2.1.2中，分别给出了现有的一些基于字节码特征的静态病毒检测技术以及基于病毒行为的动态检测技术。一般的方法是统计静态字节码或动态系统调用序列中的特征代码出现的频次[28, 14, 18, 25, 22]。然而，这样单纯统计字节码或系统调用频次得到的特征丢失了它们的次序信息。这些次序信息往往能够反映出很多病毒的规律，将其加入的病毒的特征库中，能够更准确地检测病毒。本文提出一种基于病毒字节码频次和次序信息的改进检测方法。下面分别介绍如何得到病毒这些动态及静态代码的次序及频次信息。

§3.1　静态代码分析

静态代码分析相对简单。本文中需要的二进制序列事实上即为文件的二进制流数据。这个二进制流可以使用任何一种编程语言获得。在§1.2.1.1 已经了解了如何获得二进制字节码的频次信息。在本节中，进一步统计各字节码之间的次序信息。这也正是本研究需要做的改进内容。下面给出这种次序信息的定义。

定义 3.1.1 给定文件的二进制序列，若其字节码为$Ng_1, Ng_2, \ldots Ng_L$，则每两个字节码组成的字节码对(Ng_i, Ng_j)称为次序字节码对。其中$i, j \in [1, L]$，默认Ng_i出现在Ng_j之前。这个次序对并不局限于相邻的次序，不相邻的次序对也要计算到次序信息中。

下面通过实例给出如何统计字节码次序信息。使用§1.2.1.2中的实验数据，对于其中的字节码特征，对应于每两个字节码有产生两种次序，这些次序的统计在表3.1给出。

§3.2　动态代码分析

本研究中所使用的动态代码主要指病毒对系统API函数的调用。下面简单介绍一下Windows系统的本地API。

Windows系统从WinNT到以后的版本使用两种处理器存取模式–用户模式和内核模式。一般而言，用户的应用程序运行在用户模式下，而操作系统程序运行在内核

表 3.1 部分次序字节码对的频次统计

可执行文件	字节码特征				
	$0180\|523D$	$6060\|523D$	$6060\|918A$	$0180\|E010$	$918A\|E010$
win32.a	25	4	0	19	58
win32.b	12	29	31	64	62
win32.c	18	37	50	15	40
win32.d	48	89	71	42	65
win32.e	39	73	29	107	56
win32.f	32	21	67	58	29
win32.g	29	19	82	37	48

模式下。它们的区别在于内核模式被授权了对系统内存和CPU指令存取的权限，而用户模式没有此权限。这样便能保证用户的对系统的误用或者错误不会危害整个系统的稳定。API（应用程序接口）是包括在动态链接库中的可编程的函数。它能够在用户模式或者内核模式下运行。为与通常意义下的用户模式的API函数调用，将运行在内核模式的API称为本地API[32, 33]。在Win32的子系统中，有四个用来输出API的动态链接库：User32.dll（用户接口API）、Gdi32.dll（图形接口API）、Kernel32.dl（系统管理API）和Adapi32.dll（高级系统管理API）。在WindowsXP系统下，大概有16种类的949个本地API。使用Depends.exe来查看所有Ntdll.dll输出的本地API（284个关键API），使用Ntoskrnl.exe来查看其它的API（非关键API）。

了解了API的一些内容，下面介绍程序对API函数的调用序列以及代码分析。

系统调用跟踪一个特定的过程称为序列，类似序列的集合称为序列集。研究结果发现，正常执行时的系统调用序列比较一致，而与异常执行时或者其他程序的执行相差很大[26]。因此可以使用程序对API函数调用的频次特征来区分病毒文件。本文提出的方法便是基于序列中系统调用的频次，进而得到某调用的频率特征。即将给定序列转化为系统调用频次$X_i = (f_1, f_2, \ldots f_n)$。其中$n$为可能的系统调用的总数，每个$f_i$为某个系统调用的次数[25]。然而这种只计算某个系统调用频次的处理方法，忽略了各个系统调用之间的调用次序。事实上，某些病毒文件正是改变了系统调用的次序，从而使其与正常文件产生不同。

正如静态代码分析一样，动态代码分析方法需要统计文件对系统调用的频次和次序信息。同样，有如下定义：

图 3.1 (a) 内存溢出演示; (b) 系统调用序列

定义 3.2.1 给定文件的系统调用序列，若其系统调用分别为$Sc_1, Sc_2, \ldots Sc_L$，则每两个系统调用组成的一对数据(Sc_i, Sc_j)称为次序系统调用对。其中$i, j \in [1, L]$，默认Sc_i出现在Sc_j之前。这个次序对并不局限于相邻的次序，不相邻的次序对也要计算到次序信息中。

下面通过一个小例子说明如何进行统计。

例 1 本例中，首先给出可能导致缓冲区溢出异常的一段C语言程序，如图*3.1*中的（a）所示。

程序的正常执行和异常执行产生的系统调用序列在图*3.1*中的（b）给出。

则对于系统调用的频次统计为表*3.2*。次序系统调用对的频次统计如表*3.3*。

表 3.2 系统调用的频次统计

系统调用	open	read	old_mmap	Mmap2	Munmap
正常路径频次	3	2	5	2	2
入侵路径频次	3	2	5	1	1

表 3.3 部分次序系统调用对的频次统计

次序系统调用对	$open\|read$	$read\|Munmap$	$Mmap2\|old\_map$	$Mmap2\|Munmap$
正常路径频次	6	3	0	2
入侵路径频次	6	1	0	0

上述两节给出了对病毒文件的静态字节码序列及动态系统调用序列的分析。本研究中需要的信息，一方面是这些序列中的各个字节码以及系统调用的频次信息；另一方面是字节码之间、系统调用之间的次序信息。现在明确了如何获取这些信息，将在下面具体介绍如何基于这些统计信息得到条件随机场建模所需要的特征。

§3.3 病毒信息优化采样分析

在应用代码分析技术对条件随机场建模时的主要困难在于，无论从病毒文件还是正常文件中得到的字节码信息太大。所以，对海量的文件信息进行优化采样，将字节码信息限制到一些更小范围的集合上，对病毒和正常文件的分类是十分必要的。文献[14]中，统计模块中的字节码频次信息后，直接选择频次最高的前面一部分的代码。这显然与实际情况是不吻合的。实际中，有很多病毒文件中，尤其一些零天生命周期病毒，它们当中的病毒代码甚至只出现了几次。如果按照字节码的频次去选择高频字节码，必然丢失了这部分关键病毒信息。

当前有一些特征筛选技术，最常用且有效的方法是信息增益理论[28, 12, 15]。信息增益衡量了程序中某个字节码出现的频率所占分类预测的信息数。下面我们使用该理论进程代码筛选。本文使用信息增益理论筛选字节码序列，具体介绍如下。

字节码特征筛选

定义 3.3.1 令C是一类可执行文件，C是训练数据的子集。则字节码Ng的信息增益定义为：

$$IG(Ng) = \sum_{V_{Ng}} \sum_{C} P(V_{Ng}, C) \log \frac{P(V_{Ng}, C)}{P(V_{Ng})P(C)} \tag{3.1}$$

其中C的取值有两类：病毒（V）或者正常程序（B），V_{Ng}是字节码Ng的取值，当字节码出现在程序中时$V_{Ng} = 1$；否则$V_{Ng} = 0$。$P(V_{Ng}, C)$ 是C中字节码Ng所占的比例。$P(V_{Ng}$ 是整个训练数据集中Ng所占比例。$P(C)$是属于类C的数据所占总训练数据的比例。

在本例中，将两个有次序关系的字节码对记为(Ng_i, Ng_j)。由上述定义，同理可知次序字节码对的信息增益定义为：

$$IG(Ng_i|Ng_j) = \sum_{V_{Ng_i|Ng_j}} \sum_{C} P(V_{Ng_i|Ng_j}, C) \log \frac{P(V_{Ng_i|Ng_j}, C)}{P(V_{Ng_i|Ng_j})P(C)} \tag{3.2}$$

其中的表示与上述定义一致，只是将单个字节码换成了次序字节码对。

定义 3.3.2 对于给定的字节码Ng及程序P，Ng在P中出现的频率定义为$\tau(Ng, P)$，规则化频率为$\frac{\tau(Ng,P)}{\tau}$，其中τ为$\sum_{Ng \in P} \tau(Ng, P)$。

同理，对于次序字节码对的频率定义为$\tau(Ngi|Ngj, P)$，则规则化频率为$\frac{\tau(Ngi|Ngj,P)}{\tau}$，其中$\tau$为$\sum_{Ngi|Ngj} \tau(Ngi|Ngj, P)$。

每个字节码的规则化频率一般按降序进行筛选。例如，对于共有L个字节码，M个次序字节码的文件，可以选择前L/2个字节码和前M/2个次序字节码作为实验的基础数据。为了更好的理解上述两个概念，考虑下面的例子。

例 2 假设训练数据集由*20*个可执行文件组成，其中*10*个是病毒程序，*10*个为正常程序。假设从这些程序中一共提取出*6*种字节码$Ng_i, i = 1, \ldots 6$特征。表格*3.5*和*3.4*给出了病毒程序和正常程序中对应字节码的频次。表*3.6*显示了这些字节码在病毒文件和正常文件中的集体文档频率。例如Ng_1在*10*个病毒程序中出现了两次，在*10*个正常程序中出现了*9*次（‘*1*’表示某字节码出现，‘*0*’表示不出现）。

表 3.4 病毒的字节码频次信息

病毒程序	Ng1	Ng2	Ng3	Ng4	Ng5	Ng6
v1	4	0	0	0	2	1
v2	3	3	0	0	1	2
v3	0	6	0	0	1	2
v4	0	1	0	0	2	1
v5	0	0	0	0	2	1
v6	0	0	0	5	0	0
v7	0	0	5	4	0	0
v8	0	0	6	0	0	0
v9	0	0	0	0	0	0
v10	0	0	0	0	0	0

表 3.5 正常文件的字节码频次信息

正常文件	Ng1	Ng2	Ng3	Ng4	Ng5	Ng6
b1	2	1	0	0	0	4
b2	1	1	0	2	0	3
b3	3	1	0	2	3	2
b4	1	2	1	1	7	0
b5	2	1	3	1	4	0
b6	4	3	2	0	0	0
b7	1	2	3	3	0	0
b8	1	0	1	0	0	0
b9	1	1	1	0	0	0
b10	0	0	2	0	0	0

表 3.6 字节码的集体文档频次信息

类别	值	Ng1	Ng2	Ng3	Ng4	Ng5	Ng6
V	1	2	3	1	2	5	6
V	0	8	7	9	8	5	4
B	1	9	8	7	6	3	3
B	0	1	2	3	4	7	7

使用公式(3.1)计算字节码$Ng1$的信息增益如下:

$$\begin{aligned} IG(Ng1) &= P(V_{Ng1}=1, C=V)\log\frac{P(V_{Ng1}=1, C=V)}{P(V_{Ng1}=1)P(C=V)} \\ &+ P(V_{Ng1}=0, C=V)\log\frac{P(V_{Ng1}=0, C=V)}{P(V_{Ng1}=0)P(C=V)} \\ &+ P(V_{Ng1}=1, C=B)\log\frac{P(V_{Ng1}=1, C=B)}{P(V_{Ng1}=1)P(C=B)} \\ &+ P(V_{Ng1}=0, C=B)\log\frac{P(V_{Ng1}=0, C=B)}{P(V_{Ng1}=0)P(C=B)} \\ &= \frac{2}{10}\log\frac{\frac{2}{10}}{\frac{11}{20}\cdot\frac{1}{2}} + \frac{8}{10}\log\frac{\frac{8}{10}}{\frac{9}{20}\cdot\frac{1}{2}}\frac{9}{10}\log\frac{\frac{9}{10}}{\frac{11}{20}\cdot\frac{1}{2}} \\ &+ \frac{1}{10}\log\frac{\frac{1}{10}}{\frac{9}{20}\cdot\frac{1}{2}} = 2.6568 \end{aligned} \tag{3.3}$$

使用类似方法,得到其余五个字节码的信息增益:$IG(Ng2)=2.3823, IG(Ng3)=2.5916, IG(Ng4)=2.2490, IG(Ng5)=2.0606, IG(Ng6)=2.1332$ 前四个高增益的字节码分别为$Ng1, Ng3, Ng2, Ng4$,其余两个字节码信息将被忽略。

对次序字节码对,由于其信息量大,不好一一列举,只给出部分频次信息。使用公式*3.2*计算次序字节码对应的信息增益,序字节码对应的频次信息及其信息增益在表*3.8*, *3.7*和*3.9*中给出。

表 3.7 病毒的次序字节码对的频次信息

病毒程序	$Ng1\|Ng3$	$Ng4\|Ng2$	$Ng3\|Ng5$	$Ng1\|Ng4$	$Ng5\|Ng2$	$Ng6\|Ng3$
v1	0	0	0	3	0	0
v2	0	0	0	0	2	0
v3	0	0	0	0	8	0
v4	0	0	0	16	0	0
v5	0	0	0	0	0	0
v6	0	0	0	0	0	0
v7	0	0	0	0	0	0
v8	0	0	0	0	0	0
v9	0	0	0	0	0	0
v10	0	0	0	0	0	0

表 3.8 正常文件的次序字节码对的频次信息

正常文件	$Ng1\|Ng3$	$Ng4\|Ng2$	$Ng3\|Ng5$	$Ng1\|Ng4$	$Ng5\|Ng2$	$Ng6\|Ng3$
b1	0	0	0	0	0	0
b2	0	4	0	6	0	0
b3	0	2	0	4	8	0
b4	11	7	1	3	2	0
b5	2	4	6	9	5	0
b6	5	0	0	0	0	0
b7	9	2	0	8	0	0
b8	7	0	0	0	0	0
b9	5	0	0	0	0	0
b10	0	0	0	0	0	0

表 3.9 次序字节码对的集体文档频次信息

类别	值	$Ng1\|Ng3$	$Ng4\|Ng2$	$Ng3\|Ng5$	$Ng1\|Ng4$	$Ng5\|Ng2$	$Ng6\|Ng3$
V	1	1	4	2	2	3	1
V	0	5	3	1	5	2	0
B	1	3	1	0	4	1	7
B	0	0	6	4	3	0	3
信息增益	2.019	1.010	0.892	1.121	0.087	0.762	1.431

对于测试数据集中的字节码使用由上例给出的方法，计算得到每个字节码的信息增益。然后按次序排序得到信息增益较大的一部分字节码，将其作为本研究使用的特征基本数据，具体的特征表示方法在下节介绍。对于动态的系统调用序列，采取类似的分析方法，得到信息增益较大的系统调用函数，也将其作为条件随机场建模使用的基本特征数据。

§3.4 条件随机场特征表示

上节介绍了如何将海量的字节码数据信息进行一定的筛选，得到信息增益较大的数据信息。下面将基于得到的这部分数据信息，给出条件随机场建模的特征表示。

文献[14]将文件分为相同大小的M个模块，计算每个模块中字节码序列部分频次信息的13种度量距离，然后使用决策树算法得到每个模块的信息判定（是否含有病毒代码），最后通过这些模块的有毒模块和正常模块的数量比较，得到文件是否有毒的最终判定。其中对于模块信息综合的做法并不妥当。单纯依靠数量的对比决定文件是否有毒，必然会导致很高的误报率。正如上节提到的，很多情况下，有毒模块或有毒序列出现的频次或者数量是比较少的，甚至只有几次，如果将该信息忽略，得到的判定结果必然是不准确的。

本研究使用条件随机场模型对病毒文件的字节码序列及系统调用序列的频次信息建模。由于条件随机场能够通过一些病毒数据训练模型参数，再将模型用于病毒检测，它能够反映病毒数据的本质特征。本文中仍使用文献[14]给出的13种度量距离，同时将本文的改进内容–序列之间的次序信息加入到度量计算中，得到信息的13个度量距离。这样每一种病毒文件可与一个13维的向量相对应，使用这个向量训练链式CRFs即可得到该病毒的条件随机场模型。这样每一种病毒便对应了一个条件随机场模型。对于未知的检测文件，首先提取其序列的特征信息，将其带入到各个病毒的条件随机场模型中计算条件概率，得到的条件概率最大的那个模型，便可认定该文件为此种病毒。

此外，本文还将该度量以及条件随机场的建模方法运用的动态行为分析中，对系统调用序列进行分析，同样得到序列的频次信息。下面具体介绍这13种结合了次序信息的新的数据度量方法。

1. 辛普森指标

辛普森指标是定义在生态系统中的一种度量，它衡量一个地区的物种多样性。计算机病毒系统也可称为一种特殊的生态系统，使用辛普森指标可以反映

不同病毒样本之间的差异。其表达式如下：

$$S_i = \frac{\sum n_1(n_1-1)+n_2(n_2-1)}{N_1(N_1-1)+N_2(N_2-1)} \tag{3.4}$$

其中n_1，n_2分别表示字节码和次序字节码对的频次，N_1，N_2分别表示有次序的字节码总频次。这个表达式的零值表示字节码的频次之间没有差异。类似地，随着S_i的增加，字节频次信息间的差异也会变大。

在所有子特征的定义中，用X_j表示第i个病毒或正常文件中的第j个字节码（按信息增益降序排列）的频次，Y_i表示第i个病毒或正常文件中的第j个次序对（按信息增益降序排列）的频次。对于本文使用的长度为2的字节码,j从0到65535变化。

2. 堪培拉距离

这个距离度量了一对目标的坐标之间的一系列分数差分的和。在数学上，表示为：

$$CA(i) = \sum_{j=0}^{n} \frac{|X_j - X_{j+1}| + |Y_j - Y_{j+1}|}{|X_j| + |X_{j+1}| + |Y_j| + |Y_{j+1}|} \tag{3.5}$$

3. 闵可夫斯基顺序距离

这是一个度量绝对量级差的距离表达。

$$m_i = \sqrt[\lambda]{\sum_{j=0}^{n}(|X_j - X_{j+1}|^{\lambda} + |Y_j - Y_{j+1}|^{\lambda})} \tag{3.6}$$

在本研究中，λ取3。

4. 曼哈顿距离

$$mh_i = \sum_{j=0}^{n}(|X_j - X_{j+1}| + |Y_j - Y_{j+1}|) \tag{3.7}$$

5. 契比雪夫距离

契比雪夫距离度量也被称为最大值距离。它度量了一对目标的坐标之间的差分的绝对量级：

$$ch_i = max_j\{|X_j - X_{j+1}|, |Y_j - Y_{j+1}|\} \tag{3.8}$$

6. 柯蒂斯距离

这是一个定义为字节码频次之间的绝对偏差和的标准度量距离：

$$bc_i = \frac{\sum_{j=0}^{n} |X_j - X_{j+1}| + |Y_j - Y_{j+1}|}{\sum_{j=0}^{n}(X_j + X_{j+1} + Y_j + Y_{j+1})} \tag{3.9}$$

7. 角距

这个特征对两个向量夹角的余弦值的相似度建模，这个表达式的值更大，表示两个向量更相似。

$$AS_i = \frac{\sum_{j=0}^{n}(X_j \centerdot X_{j+1} + Y_j * Y_{j+1})}{\left(\sum_{j=0}^{n}(X_j^2 + Y_j^2) \centerdot \sum_{j=0}^{n}(X_{j+1}^2 + Y_{j+1}^2)\right)^{1/2}} \tag{3.10}$$

8. 相关系数

$$\begin{aligned} CC_i = & \frac{\sum_{j=0}^{n}(X_j - X_i') \centerdot (X_{j+1} - X_i')}{\left(\sum_{j=0}^{n}(X_j - X_i')^2 \centerdot \sum_{j=0}^{n}(X_{j+1} - X_i')^2\right)^{1/2}} \\ & + \frac{\sum_{j=0}^{n}(Y_j - Y_i') \centerdot (Y_{j+1} - Y_i')}{\left(\sum_{j=0}^{n}(Y_j - Y_i')^2 \centerdot \sum_{j=0}^{n}(Y_{j+1} - Y_i')^2\right)^{1/2}} \end{aligned} \tag{3.11}$$

其中X_i', Y_i'分别表示某类文件中字节码的平均频次和次序字节码对的平均频次。

9. 熵

$$E(R) = -\sum_{i \in \Delta n} \Big(t(X_i) \log_2 t(X_i) + t(Y_i) \log_2 t(Y_i) \Big) \tag{3.12}$$

其中$t(X_i), t(Y_i)$为字节码和次序字节码对出现的频率。

10. KL发散

$$KL_i = -\sum_{j=0}^{n} \left(X_j \log \frac{X_j}{X_{j+1}} + Y_j \log \frac{Y_j}{Y_{j+1}} \right) \tag{3.13}$$

11. JS发散

$$JSD_i = \frac{1}{2}D(X_j||M) + \frac{1}{2}D(X_{j+1}||M) + \frac{1}{2}D(Y_{j+1}||N) + \frac{1}{2}D(Y_{j+1}||N) \quad (3.14)$$

其中$M = \frac{1}{2}(X_j + X_{j+1})$，$N = \frac{1}{2}(Y_j + Y_{j+1})$

12. IS发散

$$IS_i = \sum_{j=0}^{n} \left(\frac{X_j}{X_{j+1}} + \frac{Y_j}{Y_{j+1}} - \log \frac{X_j}{X_{j+1}} - \log \frac{Y_j}{Y_{j+1}} - 2 \right) \quad (3.15)$$

13. 全变差

它度量两个概率分布之间的最大可能偏差。定义为：

$$\delta_i(X_j, X_{j+1}, Y_j, Y_{j+1}) = \frac{1}{2} \sum_j \left(|X_j - X_{j+1}| + |Y_j - Y_{j+1}| \right) \quad (3.16)$$

§3.5 多引擎检测结果融合

在信息融合领域，有很多种其他不同的融合方法。如投票融合分类方法[58]，该方法通过计算对某个类或者结果集的支持票数，选择票数最多的类或结果；最大融合分类方法，它选择标准概率最高的那个类或者结果集；产品融合方法，该方法将每个类别的概率与其他所有的标准概率相乘，得到产品概率最大的类获胜。最大融合分类方法和产品融合方法对于噪声和扰动过于敏感[61]，不适合于本研究中检测引擎的病毒检测率受限于实验数据和环境的影响，存在一定扰动的情况。投票融合分类方法虽然在性能上比其他两种方法好一些[59]，却不适合于本实验中多引擎检测结果融合。受限于不同引擎的检测准确度不同，若起决定作用的大部分引擎的检测率远远低于其余部分引擎，这样得到的分类结果也是不准确的。

本研究中，我们使用D-S证据理论进行多结果融合。D-S证据理论是传统概率理论的替代方法，主要用于不确定信息的数学表示。该方法的最大改进在于它允许对集合或嵌套区间分配概率指派。它不要求对集合或区间的单个集合成分作假设。当无法获得精确的实验度量或者当信息是从经验中获得时，它是工程应用的风险和可靠性评估非常有用的工具。它一方面能够处理模糊证据，对与实际有一定偏差的数据结果有同样好的融合性能；另一方面，能够通过各个结果集自身的可靠程度，决定其在进行决策时所起的作用大小。

本研究中，各个杀毒引擎受限于自身的检测算法，对病毒的检测率通常小于1，我们可以将这个检测率作为证据理论合成时的基本置信指派，这与证据理论的特点不谋而合。而且，对于与实际情况有一定偏差的检测结果，证据理论能够对其分配较小的置信指派，使其在结果融合时起较小的作用。相比之下，文献[63]中仅将引擎之间的数目比较作为判断文件是否为病毒文件的依据，显然与实际情况不合。因此，本文选择使用D-S证据理论进行多引擎检测结果的融合工具。

§3.6 小结

本节主要介绍条件随机场建模过程中的病毒特征选择问题，首先对于海量的字节码信息，使用信息增益理论进行筛选，过滤掉一些信息增益较小的字节码信息，只选取信息增益较大的一部分字节码信息。然后在这些字节码信息的基础上，给出条件随机场的特征表示。本文采用13种信息度量来表示某类病毒文件的特征，将文件种类与这个13维向量一一对应。这样建立每种病毒对应的条件随机场模型，然后将测试数据带入每个模型中计算条件概率，最终概率最大的即为该测试文件的种类。具体实现步骤将在§4.1.1详细介绍。

第四章　多引擎云安全机制实现

§4.1　系统架构

本研究采用的系统架构主要由终端软件和云端组成[63]，如下图4.1所示。下面介绍各部分的组成及功能。

1. 终端软件，安装于终端计算机上，用于实时检测终端文件的更新。一般采用轻量级代理方式，对任何入口如USB驱动器，邮件系统附件，文件下载软件等进行实时监控。为提高时效性，文件信息一旦被写入文件系统，即将其发送至云端分析，省去了文件文件传送、分析及用户初始化的时间。当发现系统中文件更新时，终端软件对文件进行拦截和处理。这个处理主要是筛选检测引擎需要的文件信息，例如静态检测引擎所需要的字节码特征信息，动态检测引擎需要的API函数调用序列以及其他检测引擎需要的文件信息。然后，将处理结果发送至云端服务器进行检测，并接收云端服务器的检测结果，将可疑文件或病毒文件反映给用户，或者直接处理。

   终端软件的界面有三种操作模式：透明模式，报警模式和拦截模式。透明模式即终端软件的行为对终端用户来说是完全透明的。文件被发送至云端进行分析，但是文件的执行和下载不会被封锁和中断。在这种模式下，终端主机可能会被已检测到的病毒文件感染，但管理员能够利用检测警告和具体的取证信息来帮助清理被感染的系统；在报警模式和拦截模式下，文件的存取会被拦截和中断，直到终端软件接收到云端服务器对该文件的检测结果。如果该文件标记为病毒文件，则会以报警的方式通知用户。报警模式下，用户仍可以自由决定是否打开该病毒文件，而拦截模式下，终端软件将文件拦截，用户不允许打开该病毒文件。

2. 云端服务器，包括检测引擎组和结果融合单元。服务器接收到终端软件发送的可疑文件特征信息后，使用多引擎组对文件特征进行分析检测。本实验中的多引擎组主要包括静态检测引擎、动态检测引擎和常用检测引擎组。其中静态和动态检测引擎即采用第三章所述的改进方法对文件检测，常用检测引擎组选择目前流行的一些杀毒引擎，详见§4.1.1。服务器的的融合单元主要用于对多引擎检测结果进行信息融合，这里使用D-S证据理论，具体见§4.1.2。得到最终的威胁检测报告后，将其存入特征病毒库。

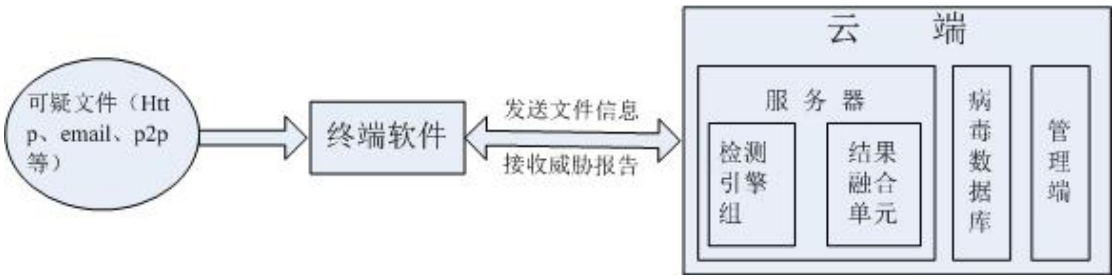

图 4.1 系统总架构

3. 云端病毒特征和行为库，用于存储病毒的特征代码段和行为等，为回溯检测提供了依据。

4. 云端管理器，用于存取病毒特征及行为库，生成病毒的威胁报告，制定并强制执行网络策略，设置网络策略违反后的报警信息，强制停用网络终端的可疑应用程序以及得到多引擎对文件的检测结果后，判定阀值的设定。其中强制停用的网络中断应用程序不一定是恶意的，他们可能对终端计算机或网络的性能有负面影响。

由上述系统架构的描述得到，对计算机上的文件进行病毒检测的流程如下图4.2所示。

§4.1.1 检测引擎

上节介绍了本研究使用的检测引擎主要由三种类型构成，其中的静态检测引擎和动态检测引擎基于本文提出的静态和动态病毒检测技术，并使用条件随机场理论进行建模，得到文件的检测结果。下面就给出他们的具体建模过程和检测流程。

静态检测引擎

在第三章中介绍了基于条件随机场的病毒特征选择。对于训练数据的每一种病毒，首先对文件的二进制码序列，统计字节码和次序字节码对的频次信息，然后对这些信息使用信息增益理论进行筛选，最后通过13中度量得到病毒对应的一个13维特征向量。本节中将基于这些特征信息建立条件随机场模型，图4.3以病毒A的检测为例，给出条件随机场模型的架构。

训练病毒库主要是指由§4.2.1中给出的病毒A的多个文件组成的数据集，在本图中是病毒A，可以使任何一种病毒或正常文件类型；

字节码及次序字节码对的信息统计主要是指对二进制文件序列中字节码频次和次序字节码对的频次计算，即使用§3.2中给出的统计方法；

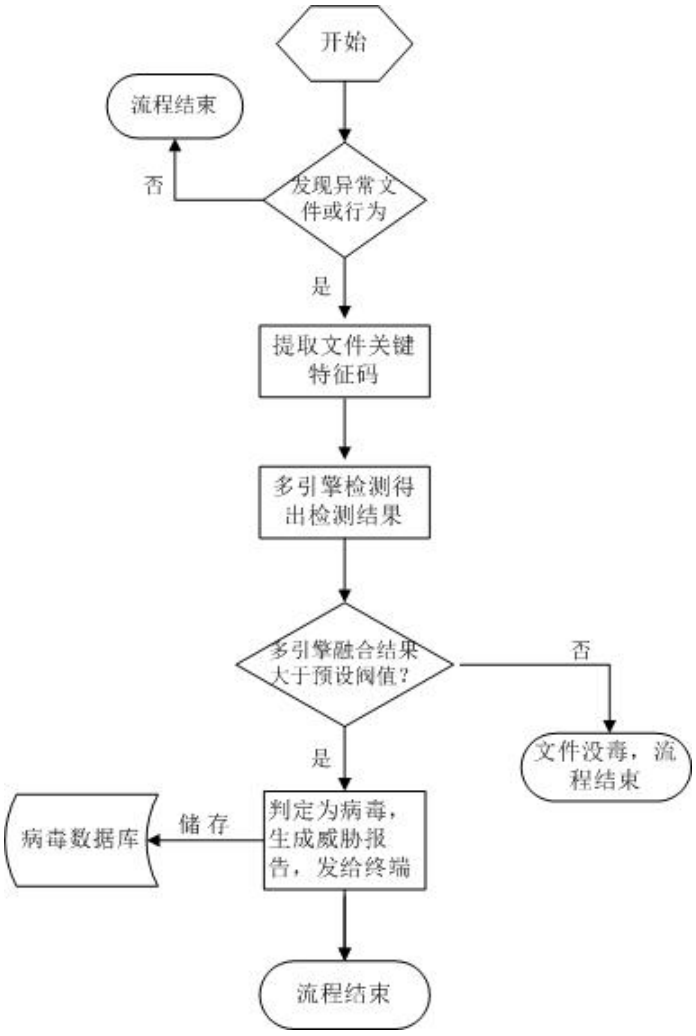

图 4.2 系统检测病毒流程

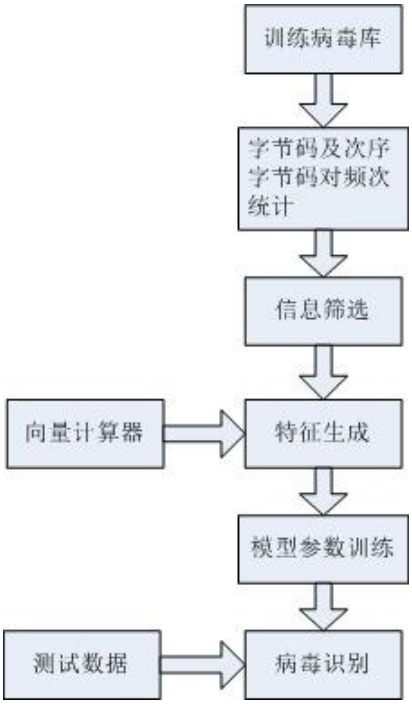

图 4.3 条件随机场模型架构

信息筛选是对已得到的字节码及次序字节码对的频次信息，使用§3.3中介绍的信息增益理论，筛选信息增益较大的字节码序列；

特征生成是指对于筛选后得到的频次统计信息，由§3.4给出的特征表示方法，使用向量计算器进行13中度量的计算，得到该病毒类型对应的13维特征向量；

模型参数训练是由得到的13维特征向量，对条件随机场模型计算该模型的特征参数，进而得到模型表示；

病毒识别即将需要进行病毒检测的测试数据，使用该模型计算病毒条件概率，再与其他模型的条件概率做大小比较。概率最大的模型的种类即为该测试文件的类型检测结果。

这样在训练数据中使用其他种类的病毒文件，便得到不同的病毒模型；在训练数据中使用正常文件类型，得到对应类型的模型。建立了这些病毒种类的条件随机场模型之后，给定测试数据，将其13维向量特征带入到每个模型中计算条件概率。概率结果最大的那个模型，即为该文件的检测结果。该结果即可作为静态检测引擎的检测结果。

动态检测引擎

相比较静态检测引擎而言，动态检测对条件随机场建模的过程大体相似。只是在第二步中，静态检测选取的是训练数据的二进制文件字节码和次序字节码对特

征，动态检测则使用系统调用和次序系统调用对的频次信息。这样针对不同类型的文件（病毒或正常文件）建立动态检测的条件随机场模型后，同样，对于测试数据，将其向量特征带入到各个模型中计算条件概率，得到概率最大的那个模型即为该文件的判定类型。在下面的结果融合单元，介绍如何将这些引擎的检测结果融合，得到最终的病毒检测报告。

§4.1.2 结果融合单元

上述内容已经介绍了静态检测和动态检测引擎的实现模型和检测机制。在图4.1中的服务器单元由多引擎检测组和融合单元组成。在本研究中，为了提高病毒检测的准确度，本文还选取了一些常用的病毒检测软件，将其检测结果最为本实验中的引擎检测结果。

由§2.2给出的证据理论，可知使用证据合成法则可以将多个证据焦元给出的结果融合。其中最关键的是通过经验等因素给出每个焦元对应的基本置信指派。对于常用的病毒检测软件，能够从网络上得到其检测的准确度，于是可以将这个准确度就作为其基本置信指派。本实验选取了AVG、Avast、F-Secure、Trend Micro和Norton这几款软件，由AV-TEST的测试结果[65]，分别计算这五种杀毒软件的查杀率，知其基本置信指派如表4.1所示。对于本文中给出的静态检测引擎和动态检测引擎，通过实验和测试数据，能够得到其检测的精度。本研究进行了这两种检测技术的实验，其检测准确度如表4.4 中的总准确率所示。这样，得到了多引擎的基本置信指派分布，由证据合成法则公式(2.23)，首先将两两引擎的检测结果融合，最终得到多检测引擎的置信指派。

表 4.1 常用五种杀毒软件的查杀率

检测软件	AVG	Avast	F-Secure	Trend Micro	Norton
查杀率/置信指派	0.83	0.75	0.92	0.75	1

此外，为了确定是否为病毒，得到某文件是病毒的置信指派后，需要设定一个阀值，当该置信指派大于阀值，则说明该文件是病毒；若小于该阀值，则认为是正常文件。阀值是对病毒误报率和准确率之间的衡量，阀值设定越高，病毒误报率越低，但病毒检测的准确率也会受到影响，可能将某些病毒文件检测为正常文件。它的值取0到1之间，取1即仅当该文件是病毒的信任度为1时，才确认为病毒文件，进行隔离或删除操作，此时的病毒误报率接近为0，但准确率可能会降低。该阀值需要多次实验或经验得出合理的设定值。云端管理器具有设定该阀值的权限。在本实验中，设定阀值为0.95

§4.2 实验及结果分析

§4.2.1 实验数据集

在本研究中，需要对病毒数据和正常文件数据建立条件随机场模型，同时要构建整个多引擎检测的云安全架构。这就使得数据集的选择，一方面能够满足条件随机场建模的需求，反映其特征规律；一方面也必须能够实现云安全架构的多引擎检测。下面介绍本研究中使用的数据集。

病毒数据集

本研究所用的病毒数据集来自于天空病毒收集网站[62]。计算机安全公司一般都有大量的病毒库，然而通常他们不会为研究者共享这些数据。所以获得一些病毒样本并不容易，尤其是最近出现的。该网站提供的是包含总计37，420个病毒样本的综合数据库。这些样本包括后门（backdoors），构造器（constructors），洪水（flooders），蝇蛆（bots），间谍软件（spyware），蠕虫（worms）以及特洛伊木马（trojans）等病毒类型。本研究中，只考虑PE格式Win32下的病毒。经过筛选后，病毒数据由616个Win32病毒样本组成。这些可执行文件主要可以分为以下六类。现在对各个类别做一下简单介绍。

后门（Backdoor）

后门是一种病毒程序，它允许对操作系统的标准授权方法进行旁路操作。由此，未经用户明确许可，便可以远程访问计算机系统。通过这种远程访问，记录和盗取用户信息便成为可能。

构造器（Constructor）

这种类别的病毒大多数包含了一些特殊的工具箱。它能够通过改变给定集的一些变量，自动创建新的病毒。

病毒（Virus）

这种病毒是一种能够复制自身，并将自身依附到其他正常程序中。当这些正常程序运行时，该病毒程序开始感染计算机。这也是最常见的病毒形式。

特洛伊（Trojan）

特洛伊是一个很宽泛的术语，它指的是一些独立的病毒程序，它们在运行一些合法功能是出现，并秘密地做一些可能对计算机有危害的活动，例如提供远程接口，数据破坏等。

蠕虫（Worm）

这种类型的病毒通过复制自身在整个计算机网络中迅速传播。

米开朗琪罗病毒（Miscellaneous）

在§1.2.1.3中提到过这种病毒。它包括拒绝服务（DoS），幽灵（nuker），开采者（exploit），黑客工具（hacktool）和洪灾（flooders）这几种类型。DoS和nuker允许攻击者在感染的计算机系统中启动病毒活动，进而可能导致拒绝服务的攻击。这些活动可能会导致速度变慢，自动重启或者是系统崩溃等。Exploit and hacktool利用经常导致内存溢出的系统缺陷。Flooder对一些有害信息流进行初始化操作，如邮件、即时信息或者短信等。

病毒的具体统计信息在表4.2中列出，该数据集的平均文件大小为64.2KB，具体大小从18字节到10M不等。一般而言，小数据病毒文件比大数据病毒更难检测。

表 4.2 病毒的统计信息

分类	数量	平均大小（千字节）	最小文件（字节）	最大文件（千字节）
Backdoor	180	257.9	56	8,602
Constructor	13	421.8	371	6,917
Virus	92	48.9	135	1,202
Trojan	121	129.7	18	5,210
Worm	115	78.9	45	2,982
Miscellaneous	95	182.3	293	10,681

正常文件数据集

本研究中所用的正常文件数据集取自于WinXP系统下system32目录中的PE格式文件，它们同样由六种不同文件类型组成：DOC（文档）、EXE（可执行程序）、JPG（图像文件）、MP3（音乐文件）、PDF（PDF文档）和ZIP（压缩文件）。这些文件类型涵盖了通常广泛使用的文件系列，从压缩文件到冗余文件，从可执行文件到文档文件。每个文件集包含有六十个典型样本，总共为360个。具体的统计信息在表4.3中列出。他们的大小从5KB到18MB不等，平均大小约为2MB。

表 4.3 正常文件的统计信息

分类	数量	平均大小（千字节）	最小文件（字节）	最大文件（千字节）
DOC	60	1,012.3	51	6,712
EXE	60	5,132.5	48	18,213
JPG	60	898.4	5	1,005
MP3	60	2,891.7	512	6,281
PDF	60	1,209.2	69	18,021
ZIP	60	7,201.7	102	10,290

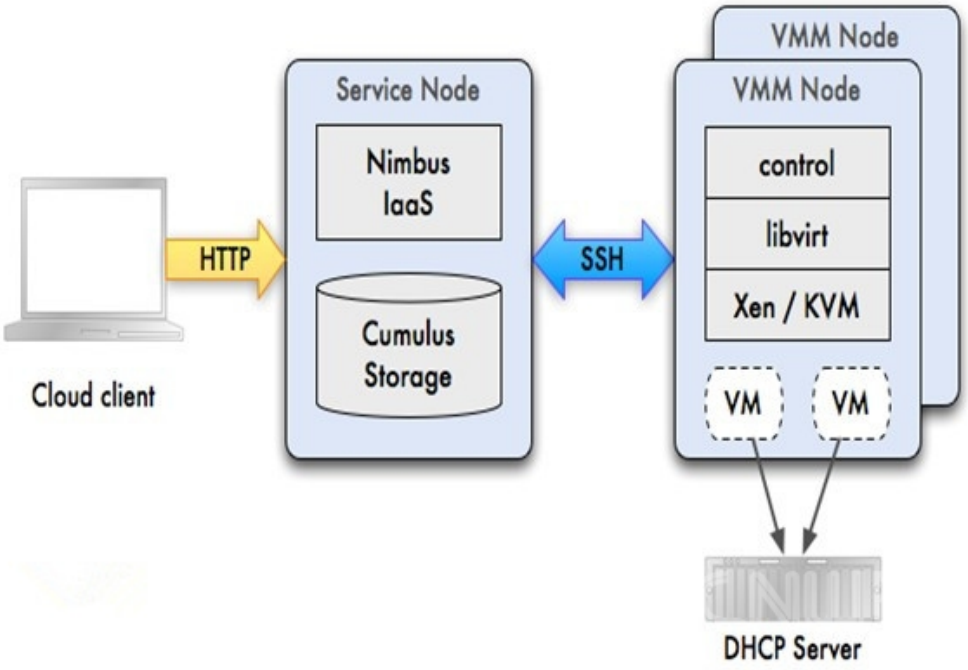

图 4.4 实验环境

§4.2.2 实验环境

由上述§4.1介绍的系统架构，各部分组成的整个实验环境如图4.4所示，下面具体介绍各个部分。

终端软件

本研究中，我们共在10台终端计算机上布置了终端软件，它们使用的操作系统有Windows XP，Windows 7及Linux 2.4。终端软件主要用于获得可执行文件的字节特征码序列和系统调用序列信息。进程生成事件或文件系统事件在进入用户终端之前，会被终端软件拦截。终端软件获取其关键信息，并送至云端服务器进行检测后，若确定其不会引发威胁事件才能够正常被执行。若发现安全威胁，则以警告或显示处理结果的方式告知用户。

本实验架构减轻了终端用户的资源消耗和负载，终端软件无论在代码大小还是在资源使用上，都设计为简单而轻量级的。如Win32的终端软件大约有1500行代码，其中60%为管理代码，这样的设计同时降低了终端软件的易受攻击性。

云端服务器和病毒库

云端服务器使用Nimbus科学计算虚拟化工具。Nimbus是一种开源云计算项目，它隶属于网格中间件。它通过一组开源工具来实现基础设施即服务（IaaS）的云计算解决方案，同时面向科学计算需求。采用部署虚拟计算机的方式，Nimbus的客户端能够租用远程资源。

云端服务器扮演了终端和后端分析引擎之间的分派管理员的角色。如图4.4所示，HTTP协议使得终端软件和网络服务之间能够通信。在网络服务内部部件之间的通信使用发布/订阅总线，它允许模块化和高度可扩展性。每个后端引擎运行在Xen虚拟容器上，提供了独立性和可扩展性的重要优势。尤其是独立性，在本实验中是非常重要的。如果某个反病毒引擎成为病毒攻击目标，并被恶意文件侵入，系统能够及时地废弃该引擎，并且不影响其他引擎的正常检测工作。

本实验选取了5个常用反病毒软件作为检测引擎：AVG、Avast、F-Secure、Trend Micro和Norton。检测得到的病毒数据库，本实验使用Cumulus云存储。它由多个模块组成，是一个能够与Amazon S3相兼容的云存储服务。

云端管理器

云端管理器提供了能够进行取证档案存取，策略强制执行，警告和威胁报告生成的管理界面。网络管理员用户通过浏览器查看这些界面，访问Nimbus服务，管理节点通过SSH和libvirt调用计算节点上的Xen或者KVM命令，并通过DHCP服务器为虚拟机分配IP地址。

§4.2.3 结果分析

本研究中，分别进行两个实验，并分析其实验结果。实验1中，首先使用静态检测引擎和动态检测引擎对§4.2.1给出的实验数据进行检测，即建立每种文件的条件随机场模型。从给出的数据库中取出100个正常文件和200个病毒文件用于条件随机场的模型训练。然后，将其余的260个正常文件和416个病毒文件作为测试数据，带入到每个模型中计算条件概率，概率最大的即为该文件的类型。

实验2中，将这些测试数据使用§4.1.2中给定的五类杀毒软件进行病毒检测，再得到一组检测结果，使用D-S证据理论进行信息融合。

检测方法的性能评价一般使用三种度量标准：准确率，检测率(TPR)和误报率(FPR)。

检出数(TP)表示被识别为病毒的病毒文件数目；

正确数(TN)表示被识别为正常文件的正常文件数目；

误报数(FP)表示被识别为病毒文件的正常文件数目；

误识数(FN)表示被识别为正常文件的病毒文件数目。

则上述三种度量标准的表达式如下：

准确率 $= \frac{(TP+TN)}{TP+TN+FP+FN}$

检测率 $= \frac{TP}{TP+FN}$

误报率 $= \frac{FP}{TN+FP}$

由实验1，得到静态检测引擎和动态检测引擎的准确率如下表4.4所示。

表 4.4 静态和动态检测引擎的准确率

种类	Backdoor	Constructor	Virus	Trojan	Worm	Miscel	正常文件	总准确率
动态	0.991	0.928	0.915	0.973	0.989	0.959	0.938	0.975
静态	0.902	0.995	0.947	0.899	0.981	0.951	0.908	0.968

由实验2中，使用常用杀毒软件对测试数据进行检测，得到的检测结果后，使用证据理论，综合表4.1和4.4中的置信指派，合成检测结果后得到总的置信指派。如对于某文件X，常用软件的检测结果如下表所示。计算得到其总的置信指派为0.972，根据阀值为0.95 可知该文件X为病毒文件。下面给出整个系统架构的检测精度，如表4.5，并在图4.5中给出其ROC曲线[66, 67]

表 4.5 总系统的检测准确率

文件种类	Backdoor	Constructor	Virus	Trojan	Worm	Miscel	正常文件	总计
准确率	0.983	0.959	0.968	0.947	0.975	0.993	0.928	0.989

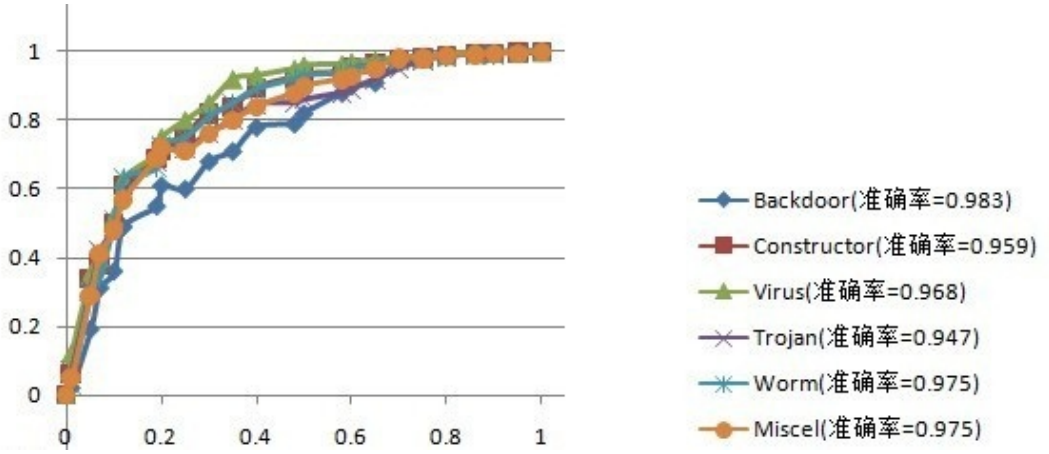

图 4.5 病毒检测结果的ROC曲线

Feature	No. of Gram/feature	AUC
F1	1	0.823
F2	1	0.839
F3	1	0.866
F4	2	0.891
F1-F2	1, 1	0.940
F1-F3	1, 1	0.928
F1-F4	1, 2	0.932
F2-F4	1, 2	0.929
F1-F2-F4	1, 1, 2	**0.962**
F3-F2	1, 1	0.954
F3-F4	1, 2	0.913
F1-F2-F3-F4	1, 1, 1, 2	0.956

图 4.6 文献[14]的检测结果

由上述结果可知，本研究提出的静态特征字节码技术和动态系统调用技术，相比较于常用的病毒查杀软件来说，其准确率有一定的提高；与文献[14]的实验数据结果（图4.6）相比，动态和静态检测引擎均有一定的提高。而本文的整个云安全机制使用多引擎检测得到的检测准确率相比动静态两个检测引擎的检测结果，以及现有的检测结果精度，又实现了一定的提高。本实验验证了本研究算法的可行性，以及其对检测准确度的改进。

总结与展望

本文提出一种基于条件随机场的云安全机制。首先通过对现有的静态和动态检测算法进行改进，得到一种利用了病毒文件字节码之间和系统调用之间次序信息的新方法。然后使用条件随机场理论对文件的向量特征建模，得到静态和动态改进方法的检测结果。最后使用D-S证据理论将动态和静态检测引擎与常用的杀毒软件的检测结果融合，得到最终的检测结果。实验表明，本文提出的病毒检测改进方法比现有的常用检测软件，在性能上有一定程度的提高。本文的云安全架构，比改进方法的检测性能有了更进一步的提高。本研究得到的检测机制能够更准确地识别病毒，为计算机用户提供更好的保护屏障。

本研究进行的实验中，数据集相对来说不够充足。在下一步的研究中，将在更大的病毒数据集上评估本研究机制。同时，也将在结构更加复杂的病毒数据集上实施本研究，期望同样获得性能的提高。

参考文献

[1] 瑞星. 瑞星2010上半年互联网安全报告[R]. http://www.rising.com.cn/about/news/rising/2010-07-30/7950_2.html.

[2] CNCERT/CNNIC. 网民网络信息安全状况报告[R]. http://tech.qq.com/zt/2010/safe/index.htm.

[3] Symantec Corporation. Symantec Internet Security Threat Report: Trends for 2010[R]. https://www4.symantec.com/mktginfo/downloads/21182883_GA_REPORT_ISTR_Main-Report_0411_HI-RES.pdf.

[4] F-Secure Corporation. F-Secure Reports: Amount of Malware Grew by 100% during 2007[R]. http://www.f-secure.com/f-secure/pressroom/news/fs_news_20071204_1_eng.html.

[5] 王瑾. 浅析云计算时代的计算机病毒防护[J]. 信息与电脑, 2010, 7: 3-4.

[6] 欧阳中辉，张晓瑜，涂帅等. “云安全”在计算机防病毒应用中的问题研究[J]. 计算机与现代化, 2010, 12: 72-75.

[7] 王跃红. 基于云安全的恶意URL动态扫描系统的设计与测试[D]. 北京：北京邮电大学，2010.

[8] 王佳, 张笈. 云安全的反病毒应用研究[J]. 信息网络安全, 2010, 7: 13-15.

[9] 张淑清. 公安信息网络防毒系统的云安全模式探析[J]. 中国人民公安大学学报（自然科学版）, 2010, 4.

[10] 张茜. 云安全环境下的恶意代码前端检测技术研究[D]. 合肥：合肥工业大学, 2011.

[11] 瑞星. 瑞星“云安全”计划白皮书[R]. http://it.rising.com.cn/new2008/News/NewsInfo/2008-07-15/1216112552d48434.shtml.

[12] Boyun Zhang, Jianping Yin, Jingbo Hao, et al. New Malicious Code Detection Based on N-Gram Analysis and Rough Set Theory[J]. Computational Intelligence and Security, 2007, 4456: 626-633.

[13] Hamid Parvin, Behrouz Minaei, Hossein Karshenas, et al. A New N-gram Feature Extraction Selection Method for Malicious Code[J]. Adaptive and Natural Computing Algorithms, 2011, 6594: 98-107.

[14] S. Momina Tabish, M. Zubair Shafiq, Muddassar Farooq. Malware Detection using Statistical Analysis of Byte-Level File Content[C]. In: Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics, New York: ACM New York, 2009: 23-31.

[15] D Krishna Sandeep Reddy, Arun K Pujari. N-gram analysis for computer virus detection[J]. Journal in Computer Virology, 2006, 2(3): 231-239.

[16] Abou-Assaleh T., Cercone N., Keselj V., Sweidan R., et al. N-gram-based Detection of New Malicious Code[C]. In: Proceedings of the 28th Annual International COMPSAC, Computer Software and Applications Conference, 2004, 2: 41-42.

[17] Damashek, M. Gauging similarity with n-grams: language independent categorization of text[J]. Science, 1995, 267(5199): 843-848.

[18] Ding Yuxin, Yuan Xuebing, Zhou Di, et al. Feature representation and selection in malicious code detection methods based on static system calls[J]. Computers & Security, 2011, 30(6-7): 514-524.

[19] Lansheng Han, Cai Fu, Deqing Zou, et al. Task-based behavior detection of illegal codes[J]. Advanced Theory and Practice for Cryptography and Future Security, 2012, 55(1-2): 80-86.

[20] Xie Han, Qiulin Tan. Dynamical behavior of computer virus on Internet[J]. Applied Mathematics and Computation, 2010, 217(6): 2520-2526.

[21] Schultz MG, Eskin E, Zadok E, Stolfo SJ. Data mining methods for detection of new malicious executables[C]. In: Proceedings of the IEEE symposium on security and privacy, Oakland CA, May 2001: 38-49.

[22] Wang M, Zheng C, Yu JJ. Native API based windows anomaly intrusion detection method using SVM[C]. In: Proceedings of the IEEE international conference on sensor networks, ubiquitous, and trustworthy computing, Taiwan; June 2006: 514-519.

[23] 王海峰, 夏洪雷, 孙冰. 基于程序行为特征的病毒检测技术与应用[J]. 计算机系统应用, 2006, 5: 29-31.

[24] Sung AH., Xu J., Chavez P., et al. Static analyzer of vicious executables[C]. In: Proceedings of the 20th annual computer security applications conference (ACSAC), Tucson, USA, Dec. 2004: 326-334.

[25] Varghese SM, Jacob KP. Process profiling using frequencies of system calls[C]. In: The second international conference on availability, reliability and security, Vienna Austria, April 2007: 473-479.

[26] Poulose Jacob K, Surekha Miriam Varghese. Anomaly Detection Using System Call Sequence Sets[J]. Journal of Software, 2007, 2(6): 14-21.

[27] T. Dube, R. Raines, G. Peterson, et al. Malware target recognition via static heuristics[J]. Computers & Security, 2012, 31(1): 137-147.

[28] Kolter J, Maloof M. Learning to detect and Classify malicious executables in the Wild[J]. Journal of Machine Learning Research, 2006, 7: 2721-2744.

[29] Igor Muttik, Chris Barton. Cloud security technologies[J]. Information Security Technical Report 2009, 14(1): 1-6.

[30] Schmidt M., Baumgartner L., Graubner, P., et al. Malware Detection and Kernel Rootkit Prevention in Cloud Computing Environments[C]. In: Parallel, Distributed and Network-Based Processing (PDP) 19th Euromicro International Conference, Ayia Napa, Feb. 9-11 2011: 603-610.

[31] Lim, S.Y., Jones, A. Network Anomaly Detection System: The State of Art of Network Behaviour Analysis [C]. In: Convergence and Hybrid Information Technology International Conference Aug. 28-30 2008: 459 - 465.

[32] Gary Nebbett Windows NT/2000 Native API Reference[M] Macmillan Technical Publishing (MTP), 2000.

[33] Sven Schreiber Undocumented Windows 2000 Secrets:A Programmer's Cookbook[M]. Boston: Addison-Wesley Longman Publishing Co. Inc., 2001.

[34] Vishwanathan, S., Schraudolph, N., et al. Accelerated training of conditional random fields with stochastic Meta-Descent[C]. In: Proc. 23rd Internat. Conf. on Machine Learning, NY: ACMNewYork, 2006: 969 – 976.

[35] Sha, F., Pereira, F. Shallow parsing with conditional random fields[C]. In: Proc. 2003 Conf. on North American Chapter of the Association for Computational Linguistics on Human Language Technology, 2003, 1: 134 – 141.

[36] Lafferty, J.D., McCallum, A., et al. Conditional random fields: Probabilistic models for segmenting and labeling sequence data[C]. In: Proc. 18th Internat. Conf. on Machine Learning, 2001: 282 – 289.

[37] Jordan, M. Learning in Graphical Models[D]. Cambridge: MIT Press, MA, 1998.

[38] Lin Peng, Zong-tian Liu, Li-min Zhang. A Recognition Approach Study on Chinese Field Term Based Mutual Information Conditional Random Fields[J]. Procedia Engineering, 2012, 29: 1365-1382.

[39] Robert Jankowski, Krzysztof Wilde. A simple method of conditional random field simulation of ground motions for long structures[J]. Engineering Structures, 2000, 22(5): 552-561.

[40] Minhua Li, Meng Bai, Chunheng Wang, et al. Conditional random field for text segmentation from images with complex background[J]. Pattern Recognition Letters, 2010, 31(14): 2295-2308.

[41] McCallum, A., Rohanimanesh, K., et al. Dynamic conditional random fields for jointly labeling multiple sequences[C]. In: NIPS03' s Workshop on Syntax, Semantics, Statistics, 2003.

[42] Sutton, C., McCallum, A., et al. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting[C]. In: Proceedings of the twenty-first international conference on Machine learning, 2004: 693 – 723.

[43] Fuchun Peng, Andrew McCallum. Information extraction from research papers using conditional random fields[J]. Information Processing & Management, 2006, 42(4): 963 – 979.

[44] Xingyu Ma, Ji Wu, Yang Shao, et al. A Hierarchical Conditional Random Field-Decision Tree Method on Chinese Person Name Recognition[J]. Energy Procedia, 2011, 13: 3229-3235.

[45] Sotirios P. Chatzis, Yiannis Demiris The echo state conditional random field model for sequential data modeling[J]. Expert Systems with Applications, 2012, 39(11): 10303-10309.

[46] Variational conditional random fields for online speaker detection and tracking[J]. Speech Communication, 2012, 54(6): 763-780.

[47] Dempster AP. Upper and lower probabilities induced by a muilti-valued mapping[J]. Classic Works of the Dempster-Shafer Theory of Belief Functions, 2008, 219: 57-72.

[48] Malcolm Beynon, Bruce Curry, Peter Morgan. The Dempster－Shafer theory of evidence an alternative approach to multicriteria decision modelling[J]. Omega, 2000, 28(1): 37-50.

[49] Mathias Bauer. Approximation algorithms and decision making in the Dempster-Shafer theory of evidence —An empirical study[J]. International Journal of Approximate Reasoning, 1997, 17(2-3): 217-237.

[50] Malcolm Beynon, Darren Cosker, David Marshal. An expert system for multi-criteria decision making using Dempster Shafer theory[J]. Expert Systems with Applications, 2001, 20(4): 357-367.

[51] 宋建勋, 张进, 吴钦章. 基于D-S证据理论的多特征数据融合算法[J]. 火力与指挥控制, 2001, 7: 96-98.

[52] Thomas Reineking. Particle filtering in the Dempster－Shafer theory[J]. International Journal of Approximate Reasoning, 2011, 52(8): 1124-1135.

[53] dddd. 基于D-S证据理论的纹理图像分类方法[J]. 西安交通大学学报, 2005, 39(8): 828-831.

[54] Mohammad Shoyaib，M. Abdullah-Al-Wadud，Oksam Chae. A skin detection approach based on the Dempster－Shafer theory of evidence[J] International Journal of Approximate Reasoning, 2012, 53(4): 636-659.

[55] Aytunc Paksoy，Mehmet G. Information fusion with dempster-shafer evidence theory for software defect prediction[J]. Procedia Computer Science, 2011, 3: 600-605.

[56] N.E.Fenton and M. Neil. A Critique of Software Defect Prediction Models[J]. EEE Transactions on Software Engineering, 1999, 25(5): 675-689.

[57] L. Guo, B. Cukic, H. Singh Predicting Fault Prone Modules by the Dempster-Shafer Belief Networks[C]. In: Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE’ 03), 2003: 249-252

[58] M. van Erp, L. Vuurpijl, and L. Schomaker An overview and comparison of voting methods for pattern recognition[C]. In: Proc. of the 8th IWFHR, 2002: 195- 200.

[59] Robert P. W. Duin and David M. J. Tax Experiments with Classifier Combining Rules[J]. Multiple Classifier Systems, 2000, 1857, 16-29.

[60] Sentz K, S Ferson. Combination of evidence in Dempster－Shafer theory[R]. SAND2002-0835 Technical Report, New Mexico.

[61] R.P.W. DUIN, D.M.J. TAX. Experiments with Classifier Combining Rules[J]. Lecture Notes in Computer Science, 2000, 1857: 16-29.

[62] VX Heavens Virus Collection. VX Heavens website[R]. http://vx.netlux.org

[63] J. Oberheide, E. Cooke, F. Jahanian. CloudAV: N-version antivirus in the Network Cloud[C]. In: Proceedings of the 17th conference on Security symposium, 2008: 91-106.

[64] Algirdas Avizienis. The n-version approach to fault-tolerant software[J]. IEEE Transactions on Software Engineering, 1985, SE-11(12): 1491 - 1501.

[65] AV-TEST. Test reports of AV-TEST[R]. http://www.av-test.org/en/tests/test-reports/janfeb-2012.

[66] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers[R]. TR HPL-2003-4, 2004, http://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf.

[67] S.D. Walter. The partial area under the summary ROC curve[J]. Statistics in Medicine, 2005, 24(13): 2025-2040.

致　谢

首先，衷心感谢我的导师李明军老师和程戈老师。尤其是程戈老师，在我迷茫无助的时候，给予我学习的动力和方向，让我有决心走上计算机科研的道路。本文在程老师的严格要求和精心指导下完成。从最初的选题、资料的收集、搭建实验平台以及文章的结构安排到后面的写作工作都凝聚着恩师的心血。同时，在我的学习和生活中，程老师也给予我精心的指导和细心的关怀。在学习上，程老师渊博的学识，开阔的学术视野，活跃的学术思想，严谨的治学态度，给我留下了深刻的印象，令人由衷地敬佩。在生活上，程老师同样给予了无微不至的关怀，教会我要有持之以恒和专注的精神，这些都将是我人生中最宝贵的财富。在以后的学习和生活中，我会刻苦努力，继续拼搏，争取更多的科研机会并努力取得更大的成果。

其次，衷心感谢我的同门王向阳，王丽；我的师弟李浩，曹利振，师妹徐双艳。虽然与大家在一起的时间不长，但有你们一起学习和陪伴的日子是充实快乐的，也感谢你们对我的帮助和支持。

另外，衷心感谢数学与计算科学学院的所有老师，谢谢你们给我们提供了全校最好的学习和科研环境。

最后，我要特别感谢我的父母、亲友，谢谢他们给予我的支持和理解。

基于条件随机场的多引擎云安全机制研究

作者： 杜亚娟
学位授予单位： 湘潭大学

参考文献(7条)

1. 欧阳中辉,张晓瑜,涂帅,顾佼佼 "云安全"在计算机防病毒应用中的问题研究[期刊论文]-计算机与现代化 2010(12)

2. 王跃红 基于云安全的恶意URL动态扫描系统的设计与测试[学位论文]硕士 2010

3. 王佳,张笈 云安全的反病毒应用研究[期刊论文]-信息网络安全 2011(07)

4. 张淑清 公安信息网络防毒系统的"云安全"模式探析[期刊论文]-中国人民公安大学学报（自然科学版） 2010(04)

5. 张茜 云安全环境下的恶意代码前端检测技术研究[学位论文]硕士 2011

6. 王海峰,夏洪雷,孙冰 基于程序行为特征的病毒检测技术与应用[期刊论文]-计算机系统应用 2006(05)

7. 宋建勋,张进,吴钦章 基于D-S证据理论的多特征数据融合算法[期刊论文]-火力与指挥控制 2010(07)