基于聚类分析的内核恶意软件特征选择

陈志锋* 李清宝 张 平 冯培钧
(解放军信息工程大学 郑州 450001)
(数学工程与先进计算国家重点实验室 郑州 450001)

摘 要：针对现有基于数据特征的内核恶意软件检测方法存在随特征的增多效率较低的问题，该文提出一种基于层次聚类的特征选择方法。首先，分析相似度计算方法应用于数据特征相似度计算时存在的困难，提出最长公共子集并设计两轮 Hash 求解法计算最长公共子集；其次，设计基于最长公共子集的层次聚类算法，有效地将相似特征聚类成簇；在此基础上，设计基于不一致系数的内核恶意软件特征选择算法，大大减少特征数，提高检测效率。实验结果验证了方法的有效性，且时间开销在可接受的范围内。

关键词：数据特征；最长公共子集；层次聚类；特征选择；内核恶意软件
中图分类号：TP316; TP309 **文献标识码**：A **文章编号**：1009-5896(2015)12-2821-09
DOI: 10.11999/JEIT150387

Signature Selection for Kernel Malware Based on Cluster Analysis

Chen Zhi-feng Li Qing-bao Zhang Ping Feng Pei-jun
(*PLA Information Engineering University*, *Zhengzhou* 450001, *China*)
(*State Key Laboratory of Mathematical Engineering and Advanced Computing*, *Zhengzhou* 450001, *China*)

Abstract: As current kernel malware detection method based on data signature exists the problem that its efficiency decreases with the growth of the number of signatures, a signature selection method for kernel malware based on hierarchical cluster is presented. First, since current similarity calculation methods are difficult to be applied to data signature selection, a longest common subset based method and a 2-round Hash computation algorithm are introduced. Second, a longest common subset based hierarchical cluster algorithm is presented, thereby performing similar signature aggregation effectively. Finally, a signature selection algorithm based on inconsistent coefficient is designed to reduce the number of signatures. Experimental results show the effectiveness of the method, and performance evaluations indicate that algorithm runtime is acceptable.

Key words: Data signature; Longest common subset; Hierarchical cluster; Signature selection; Kernel malware

1 引言

随着计算机和互联网技术的快速发展，恶意软件对计算机的危害性日益加重，内核恶意软件是针对内核进行攻击的恶意程序，对计算机系统造成的危害更底层、更彻底，攻击具有隐蔽性、持久性等特点。内核恶意软件检测已成为重要的研究方向之一。

现有的内核恶意软件检测方法可分为基于启发式的检测和基于特征的检测两大类[1]。其中基于特征的检测方法是当前的主流方法。特征的描述能力决定了基于特征的检测方法能否有效检测内核恶意软件的能力[2]。

传统的恶意代码特征大多使用代码特征和序列特征描述法。代码特征主要是指静态二进制特征，其本质上是一段包含汇编指令等信息的二进制字符串。该特征对于已知的恶意软件具有较好的效果，对于未知的恶意软件无能为力。文献[3~5]将恶意代码映射到灰度图片，通过对图片进行分块建立恶意代码特征并通过特征相似性匹配进行恶意代码检测。该方法能够检测一些未知的恶意软件及变种，但对于加壳的恶意软件效果较差。序列特征描述法主要采用系统调用或者指令序列描述特征。如，Ding 等人[6]将特征描述为基于控制流的操作码序列，Wang 等人[7]用系统调用数目描述特征。序列描述法针对代码或行为的先后次序，易受代码混淆手段的干扰。常用的还有控制流程图(Control Flow Graph,

收稿日期：2015-04-02；改回日期：2015-07-30；网络出版：2015-10-16
*通信作者：陈志锋 516975104@qq.com
基金项目：核高基国家科技重大专项(2013JH00103)和国家 863 计划目标导向项目(2009AA01Z434)
Foundation Items: The National Science and Technology Major Project of China (2013JH00103); The National 863 Program of China (2009AA01Z434)

CFG)描述法[8]。它以代码的执行流程描述特征，但因局限于代码执行顺序，易受顺序无关操作调换等混淆方法的干扰，适合于合法软件的完整性检测。为了克服这些特征描述方法存在的问题，研究人员针对内核恶意软件的攻击对象，提出了内核数据不变量特征[9,10]，基于该类特征能够检测一些未知的恶意软件。但是并不是所有的内核数据都具备不变量特征，一旦攻击者攻击这类数据，这种方法也将失效。为此，文献[11]根据内核数据访问的一般性质，提出了数据特征。该特征充分描述了恶意软件运行过程中对内核数据的篡改行为，能够较好地用于内核恶意软件检测。但文献[11]通过特征匹配实现基于数据特征的检测方法导致检测效率随着特征数量的增多而大幅度降低。

针对基于数据特征的内核恶意软件检测方法存在的问题，本文提出了基于聚类分析的恶意软件特征选择方法。该方法在分析数据特征的组成和相似性计算方法不足的基础上提出了最长公共子集(Longest Common Subset, LCS)的概念，并讨论了基于最长公共子集的相似度计算方法。然后阐述了数据特征层次聚类和特征选择过程，并给出了基于选择特征的恶意软件检测方法。最后通过实际应用中的内核恶意软件样本对该方法进行功能和性能评估。

2 数据特征

2.1 数据特征描述方法

为了更好地说明基于数据特征的聚类分析以及特征选择方法，本文首先对文献[11]提出的数据特征作简要介绍。

定义 1[11] DBE(Data Behavior Element)是一个 5 元式 (c,o,m,i,f)，其中，c 为访问内核数据的代码；o 为访问内核数据的操作，包括读或写，$o=1$ 表示写操作，$o=0$ 表示读操作；m 为数据对象类型，包括静态数据或动态数据，$m=1$ 表示动态数据，$m=0$ 表示静态数据；i 为数据对象标识，当 $t=0$ 时，i 是基于编译阶段的符号信息赋予的编号；当 $t=1$ 时，i 是分配该数据的分配位置和数据类型；f 为被访问的数据对象字段的偏移值。

根据定义 1，一个 DBE 给出了访问一个内核数据的过程。操作系统内核中包含有成百上千种数据类型，这些数据类型又定义了成千上万个内核数据。因此，所有的 DBE 集合描述了内核一次运行过程的数据访问情况，详见定义 2。

定义 2[11] 内核一次运行实例的数据访问可由序列 $D_m=\{\mathrm{DBE}_i,\mathrm{DBE}_{i+1},\cdots,\mathrm{DBE}_j\}$ $(j>i,i\geq 0)$ 定义，其中 m 表示内核的某一次运行。

由定义 2 可知 D_m 给出了内核从开机到关机这一过程的所有内核数据访问模式，描述了内核的某一次运行过程中所有内核数据在生命周期内的使用情况。通过分析良性内核和含恶意软件运行的内核的数据访问模式差异，即可构建恶意软件数据特征，从而可以基于该特征检测恶意软件。

假设内核恶意软件 M 在第 i 次运行中对应的数据访问序列是 $D_{\mathrm{M},j}$，不含恶意软件的内核在第 j 次运行中对应的数据访问序列为 $D_{B,j}$，那么对 k 次恶意软件运行和 l 次良性内核运行收集的数据访问序列应用集合操作得到内核恶意软件 M 的数据特征为

$$S_{\mathrm{M}}=\bigcap_{i\in[1,k]}D_{\mathrm{M},i}-\bigcup_{j\in[1,l]}D_{B,j} \tag{1}$$

式(1)中 S_{M} 表示在 k 次内核恶意软件运行中均出现，但在 l 次良好内核运行中从未出现的数据访问序列集合。

2.2 问题分析

前文已指出，随着内核恶意软件样本的增多，数据特征库也随之增大，这将导致基于数据特征的检测效率大打折扣。据统计，新出现的恶意代码大部分是在原恶意代码基础上修改转换而来[12]，它们对内核对象的篡改行为本质上不会发生变化。故通过分析数据特征相似性，从相似的特征中选取代表特征，减少特征数，提高检测效率。

聚类是按照事物的某些属性，把事物聚集成簇，使簇内的对象之间具有较高的相似性，而不同簇的对象之间的相似度较差[13,14]。因此，将聚类分析技术应用于数据特征选择能够解决所述问题。

根据提取特征模型的差异，特征相似性计算方法主要包括余弦相似度、欧氏距离、Jaccard 系数、最长公共子序列(Longest Common SubseQuent, LCSQ)等[15]。不同的计算方法使用的场景不同。余弦相似度、欧氏距离主要用于向量化特征的距离计算，要求特征具有方向和长度，适用于线性的特征模型。编辑距离、最长公共子串(Longest Common SubsTring, LCST)适用于字符串相似性比较，编辑距离通过统计删除、插入、替换操作的次数计算距离，最长公共子串计算过程中要求子串在源串中连续；Jaccard 系数考虑了两个样本特征的交集和并集，适用于特征项定义明确的样本相似性计算；最长公共子序列适用于具有先后顺序的特征相似性计算，譬如系统调用序列等；图相似性计算一般采用图匹配技术实现，适用于特征元素之间具有相互关系的特征相似性计算。

根据数据特征模型可知，数据特征是由五元式

项组成的集合，该集合中的每一项元素之间不存在必然的先后顺序。譬如对于系统调用表篡改操作，攻击者修改了系统调用表表项的 3 个入口地址，这 3 个操作先后顺序不影响攻击者实现攻击。它不同于向量化特征，可以没有方向，长度动态变化，故余弦相似度、欧氏距离不适用于数据特征相似度计算；并且特征中没有重复元素存在，元素之间并没有先后顺序，故编辑距离、最长公共子序列、最长公共子串也不适用。进一步分析数据特征项，五元式中前 4 个属性取值唯一，最后一个属性因偏移值存在单一取值和多值两种情况，导致即使特征项不一样也可以认为它们是同一个特征项。譬如(η, 1, 0, 0xffffffff8180320,624)和(η, 1, 0, 0xffffffff8180320, {120,624})，由于 624∈{120,624}，且元素的前 4 项完全一样，所以我们认为(η, 1, 0, 0xffffffff8180320, 624)和(η, 1, 0, 0xffffffff8180320,{120,624})是同一个特征项。因此，对于数据特征，计算它们的相似性不能单纯考虑完全一样的特征项，还需考虑特征项的偏移关系。针对这些问题，借鉴最长公共子序列和最长公共子串，我们提出了最长公共子集，用于计算数据特征的相似性。

3　最长公共子集

根据上节的问题分析，数据特征相似性分析时必须将特征的每一项分为两部分进行处理，那么最长公共子集可定义为：

定义 3　对于数据特征 S_1, S_2，它们的最长公共子集 $\mathrm{LCS}(S_1,S_2)=\{S_1 \cap S_2, S_{\mathrm{other}}\}$，其中 $S_1 \cap S_2$ 是特征项完全一致的公共部分，S_{other} 是 S_1 中元素与 S_2 中元素的前 4 项均一致且偏移值项存在属于或者包含关系的元素。

借鉴最长公共子串和最长公共子序列的定义，我们给出了最长公共子集距离度量函数的定义，见式(2)。

$$\left.\begin{aligned}&\mathrm{LCS}:\sum^*\times\sum^*\to\sum^*\\&d(S_1,S_2)=|S_1|+|S_2|-2|\mathrm{LCS}(S_1,S_2)|\end{aligned}\right\}\quad(2)$$

定理 1　$d(S_1,S_2)$ 是 $\sum^*$ 的度量函数。

证明　根据距离度量方法的定义，要证明 $d(S_1,S_2)$ 是一个度量函数，必须证明该函数满足以下度量属性。

(1)非负性 $d(S_1,S_2)\ge 0$：　由于 $\mathrm{LCS}(S_1,S_2)\subseteq S_1$，$\mathrm{LCS}(S_1,S_2)\subseteq S_2$，所以有 $|\mathrm{LCS}(S_1,S_2)|\le|S_1|$，$|\mathrm{LCS}(S_1,S_2)|\le|S_2|$。那么 $2|\mathrm{LCS}(S_1,S_2)|\le|S_1|+|S_2|\Rightarrow 0\le|S_1|+|S_2|-2|\mathrm{LCS}(S_1,S_2)|$，即 $0\le d(S_1,S_2)$。

(2)同一性 $d(S_1,S_2)=0\Leftrightarrow S_1=S_2$：　$\Leftarrow$：如果 $S_1=S_2=S$，那么 $\mathrm{LCS}(S_1,S_2)=S$，故 $d(S_1,S_2)=|S_1|+|S_2|-2|\mathrm{LCS}(S_1,S_2)|=|S|+|S|-2|S|=0$。$\Rightarrow$：如果 $d(S_1,S_2)=0$，那么 $|S_1|+|S_2|=2|\mathrm{LCS}(S_1,S_2)|$。由于 $|\mathrm{LCS}(S_1,S_2)|\le|S_1|$，$|\mathrm{LCS}(S_1,S_2)|\le|S_2|$，所以 $2|\mathrm{LCS}(S_1,S_2)|\le|S_1|+|S_2|$，又由于 $|S_1|+|S_2|=2|\mathrm{LCS}(S_1,S_2)|$，所以 $|\mathrm{LCS}(S_1,S_2)|=|S_1|=|S_2|$。又 $\mathrm{LCS}(S_1,S_2)\subseteq S_1$，故可得 $\mathrm{LCS}(S_1,S_2)=S_1$。同理可得 $\mathrm{LCS}(S_1,S_2)=S_2$。从而，$S_1=S_2$。

(3)对称性 $d(S_1,S_2)=d(S_2,S_1)$：　由 $\mathrm{LCS}(S_1,S_2)=\mathrm{LCS}(S_2,S_1)$ 得 $|S_1|+|S_2|-2|\mathrm{LCS}(S_1,S_2)|=|S_2|+|S_1|-2|\mathrm{LCS}(S_2,S_1)|$，即 $d(S_1,S_2)=d(S_2,S_1)$。

(4)三角不等式 $d(S_1,S_2)+d(S_2,S_3)\ge d(S_1,S_3)$：根据 $d(S_1,S_2)$ 的定义，该式可扩展为

$$\begin{aligned}&|S_1|+|S_2|-2|\mathrm{LCS}(S_1,S_2)|+|S_2|+|S_3|\\&-2|\mathrm{LCS}(S_2,S_3)|\ge|S_1|+|S_3|-2|\mathrm{LCS}(S_1,S_3)|\\&\Rightarrow|S_2|+|\mathrm{LCS}(S_1,S_3)|\ge|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|\end{aligned}$$

故只需证明 $|S_2|+|\mathrm{LCS}(S_1,S_3)|\ge|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|$ 即可。

由于 $\mathrm{LCS}(S_1,S_2)\subseteq S_2, \mathrm{LCS}(S_2,S_3)\subseteq S_2$，那么根据 $\mathrm{LCS}(S_1,S_2)$ 和 $\mathrm{LCS}(S_2,S_3)$ 的长度和位置，S_2, $\mathrm{LCS}(S_1,S_2)$ 和 $\mathrm{LCS}(S_2,S_3)$ 之间的关系有 6 种情况，但 $\mathrm{LCS}(S_1,S_2)$ 和 $\mathrm{LCS}(S_2,S_3)$ 可相互交换，故只需考虑 3 种情况即可。不妨设 S_2 包含 4 个元素，即 $S_2=\{A_1,A_2,A_3,A_4\}$，那么 3 种情况如图 1 所示。

如图 1(a)，$\mathrm{LCS}(S_1,S_2)=\{A_1,A_2\}$，$\mathrm{LCS}(S_2,S_3)=\{A_4\}$，那么 $|S_2|=4, |\mathrm{LCS}(S_1,S_2)|=2, |\mathrm{LCS}(S_2,S_3)|=1$，$|\mathrm{LCS}(S_1,S_3)|\ge 0$。从而 $|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|=3$。故 $|S_2|+|\mathrm{LCS}(S_1,S_3)|\ge 4>|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,$

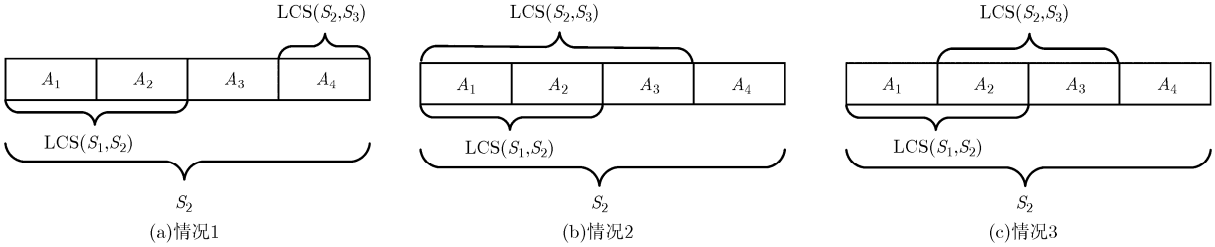

图 1 特征的最长公共子集关系

$S_3)|$，即 $|S_2|+|\mathrm{LCS}(S_1,S_3)|\geq|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|$。

如图 1(b), $\mathrm{LCS}(S_1,S_2)=\{A_1,A_2\}$，$\mathrm{LCS}(S_2,S_3)=\{A_1,A_2,A_3\}$，那么 $|S_2|=4$，$|\mathrm{LCS}(S_1,S_2)|=2$，$|\mathrm{LCS}(S_2,S_3)|=3$，则 $|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|=5$。

又 $\{A_1,A_2\}\subseteq\mathrm{LCS}(S_1,S_2)$,$\{A_1,A_2\}\subseteq\mathrm{LCS}(S_2,S_3)$，且 $\mathrm{LCS}(S_1,S_2)\subseteq S_1$，$\mathrm{LCS}(S_2,S_3)\subseteq S_3$，故 $\{A_1,A_2\}\subseteq S_1$,$\{A_1,A_2\}\subseteq S_3$。因此，$\mathrm{LCS}(S_1,S_3)$ 至少包含两个元素 A_1,A_2，即 $|\mathrm{LCS}(S_1,S_3)|\geq 2$。因此，$|S_2|+|\mathrm{LCS}(S_1,S_3)|\geq 4+2=6>|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|$，即 $|S_2|+|\mathrm{LCS}(S_1,S_3)|\geq|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|$。

如图 1(c), $\mathrm{LCS}(S_1,S_2)=\{A_1,A_2\}$，$\mathrm{LCS}(S_2,S_3)=\{A_2,A_3\}$，那么 $|S_2|=4$,$|\mathrm{LCS}(S_1,S_2)|=2$，$|\mathrm{LCS}(S_2,S_3)|=2$，则 $|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|=4$。又 $A_2\in\mathrm{LCS}(S_1,S_2)$ 且 $A_2\in\mathrm{LCS}(S_2,S_3)$，即 $\{A_2\}\subseteq\mathrm{LCS}(S_1,S_2)$,$\{A_2\}\subseteq\mathrm{LCS}(S_2,S_3)$。又 $\mathrm{LCS}(S_1,S_2)\subseteq S_1$,$\mathrm{LCS}(S_2,S_3)\subseteq S_3$，故有 $\{A_2\}\subseteq S_1$,$\{A_2\}\subseteq S_3$。因此，$\mathrm{LCS}(S_1,S_3)$ 至少包含一个元素 A_2，即 $|\mathrm{LCS}(S_1,S_3)|\geq 1$，那么 $|S_2|+|\mathrm{LCS}(S_1,S_3)|\geq 4+1=5>(4=|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|)$,即 $|S_2|+|\mathrm{LCS}(S_1,S_3)|\geq|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|$。

综上可得，$|S_2|+|\mathrm{LCS}(S_1,S_3)|\geq|\mathrm{LCS}(S_1,S_2)|+|\mathrm{LCS}(S_2,S_3)|$，即 $d(S_1,S_2)+d(S_2,S_3)\geq d(S_1,S_3)$。证毕

根据度量函数，特征相似性函数定义为

$$\left.\begin{aligned}&\mathrm{sim}:\sum^*\times\sum^*\to[0,1]\\&\mathrm{sim}(S_1,S_2)=1-\frac{d(S_1,S_2)}{|S_1|+|S_2|}=\frac{2|\mathrm{LCS}(S_1,S_2)|}{|S_1|+|S_2|}\end{aligned}\right\}\quad(3)$$

如果每个特征的长度均一致(对于长度不一致的特征，可以加入非特征元素使得两个特征长度一致)，不妨设长度为 λ，那么相似性函数式(3)可简化为

$$\left.\begin{aligned}&\mathrm{sim}':\sum^*\times\sum^*\to[0,1]\\&\mathrm{sim}'(S_1,S_2)=1-\frac{d(S_1,S_2)}{|S_1|+|S_2|}=\frac{|\mathrm{LCS}(S_1,S_2)|}{\lambda}\end{aligned}\right\}\quad(4)$$

从式(4)可以看出，相似性取值只与最长公共子集相关。

4 基于最长公共子集的特征选择方法

4.1 求解最长公共子集

根据最长公共子集的定义，其值包括两部分，一部分是两个特征交集元素的总数，另一部分是特征元素的偏移部分满足给定关系的元素个数。因此，在求解最长公共子集时需要分别计算这两部分的值。

现有的交集求解主要有两种方法，一种是求解集合的所有子集，然后比较子集的一致性；另一种是双重循环遍历法，对两个集合的所有元素进行一一比较，若相同则加入一个新的集合中，最终新集合中的所有元素构成交集。

假设数据特征的长度为 n，那么第 1 种方法的时间复杂度为 $O(2^n)$，第 2 种方法的时间复杂度为 $O(n^2)$。对于复杂的样本特征，它们的开销较大。为此，本文提出了两轮 Hash 求解法，依据 Hash 表中元素的唯一性实现最长公共子集的求解，每一轮求解一部分值，具体过程如表 1 所示的算法 1。

不妨设 S_1，S_2 的特征长度为 n，那么根据算法

表 1 最长公共子集求解算法

算法1 LCS(S_1 , S_2)
输入：特征 S_1 , S_2
输出：最长公共子集的长度
(1) list<string>list$_1$=new list<string>//存储特征 S_1 中的元素
(2)list<string>list$_2$=new list<string>//存储特征 S_2 中的元素
(3) list<string>list$_3$ = new list<string>//存储最长公共子集
(4)n=0
(5)Hashset<string>hashset=new Hashset<string>//用于Hash计算时存储
(6) **for** each item in S_1
(7) list$_1$.add(item)
(8) **for** each item in S_2
(9) list$_2$.add(item)
(10)**for** each item in list$_1$
(11) hashset.add(item)
(12)**for** each item in list$_2$
(13) **if** hashset.add(item)==false then
(14) list$_3$.add(item)
(15) list$_1$.remove(item)//去除子集
(16) list$_2$.remove(item)
(17) n++
(18)hashset.reset()//清空hashset，用于下一轮计算
(19)**for** each item in list$_1$
(20) hashset.add(item.remove(item.of))
(21)**for** each item in list$_2$
(22) **if** hashset.add(item.remove(item.of))==false
(23) item$_1$=select(item,list$_1$)//选择冲突项
(24) **if** item.f̂item$_1$.f then
(25) n++
(26) **if** item.f̂ item$_1$.f then
(27) n++
(28)return n

1 可知，算法的第 6～17 行为第 1 轮 Hash 求解最长公共子集的第 1 部分组成，其中第 6, 7 行、第 8, 9 行分别对特征进行预处理，各需要 $O(n)$ 的时间，第 10, 11 行是求解特征 S_1 的 Hash 表，需要 $O(n)$ 的时间，第 12～17 行求解特征 S_1 和 S_2 的子集，需要 $O(n)$ 的时间；算法的第 19～27 行为第 2 轮 Hash 计算，主要用于特殊处理特征元素中的偏移值项，其中第 19, 20 行处理去除偏移值项的特征 S_1，需要 $O(n)$ 的时间，第 21～27 行进行偏移值关系比较，需要 $O(n)$ 的时间。因此，本算法的时间复杂度为 $O(n)$。该时间复杂度为线性时间复杂度，仅与特征长度相关，相较于上述两种方法大大降低了求解最长公共子集的时间开销。

4.2 基于最长公共子集的聚类算法

本节将介绍如何利用最长公共子集和相似度函数进行特征聚类。据不完全统计，内核恶意软件仅占恶意软件的 4%，数量相对较少。为此，本文选择了适用于规模数较小的层次聚类算法[16]。按层次聚类分析中的合并实现方式，将特征样本根据层次结构方式进行合并，直到终止条件满足为止。这里我们选择自底向上的合并方式，并采用类间平均聚合方法 AL 计算新生成的类与各个旧类之间的相似度。

假设包含样本特征集合 $S=\{s_1,s_2,\cdots,s_n\}$ 共有 n 个特征，阈值为 θ，聚类过程如表 2 所示的算法 2。

由于算法 2 中在聚类时加入了阈值判断，故最好情况下只需进行 1 轮相似度计算，最坏情况下需要进行 $n-1$ 轮相似度计算。但是不管是 1 轮还是 $n-1$ 轮相似度计算，时间复杂度均为 n^2 量级，只是量级系数不一样而已。总的来说，由于内核恶意软件样本数远小于应用层恶意软件，故聚类分析所需的时间可接受。

4.3 特征选择

算法 2 将特征集最终构成一棵分层聚类树，下面对聚类树进行划分，然后从划分类中选择特征代表。文献[17]指出，对聚类树进行划分时存在聚类数量和聚类规模的权衡问题，也就是“一致性”问题。本文借鉴不一致系数[17]对聚类树进行剪枝，对聚类树进行划分，确定最终的分类个数。在聚类过程中，若某一次聚类所对应的不一致系数较上次有较大幅度的增加，则表明该次聚类效果较差，而上次的聚类效果较好。增加的幅度越大，上次的聚类效果就越好。

不一致系数的计算时需要考虑当前链接处向下的深度，即参与计算涉及的链接的层数。假设经算法 2 计算得到的聚类树用数组 $Z[m-1][4]$ 存储，其中 m 为特征样本数，$(S_i,S_j,S_{ij},\mathrm{sim}_{ij})$ 是某一行的取值，S_i, S_j 是聚类簇的编号，S_{ij} 是聚类后簇的编号，sim_{ij} 是两个簇的相似度；$M(k)$ 表示第 k 次聚类时涉及的所有链接长度(即聚类相似度)的均值；$\mathrm{SD}(k)$ 表示第 k 次聚类时涉及的所有链接长度的标准差，那么对于第 k 次聚类得到的链接，不一致系数为

$$\mathrm{IC}(k)=\frac{Z[k][4]-M(k)}{\mathrm{SD}(k)} \tag{5}$$

对于聚类树的叶子节点，由于它们向下没有其它节点，故当它们聚类时，对应的不一致系数为 0。非叶子节点处的聚类，不一致系数按照式(5)计算。

表 2 基于 LCS 的聚类算法

```
算法2  聚类算法LCS-clustering
输入：特征集S，阈值
输出：特征集nodes
(1) nodes=make-set(S)//将每一个样本特征归为一类
(2) while len(nodes)>1
(3)   max_sim=0//存储最大相似度值
(4)   max_pair = (−1,−1) //最大相似度值对应的两个样本特征
(5)  for i in range(len(nodes))
(6)    for j in range(i+1, len(nodes))
(7)      if len(nodes[i])==1 && len(nodes[j])==1 then
(8)        d=|LCS(nodes[i], nodes[j])| //最长公共子集
(9)        sim_ij=2d/(|nodes[i]|+|nodes[j]|)//相似度计算
(10)         if sim_ij ≥ max_sim then
(11)           max_sim=sim_ij
(12)           max_pair = (i, j)
(13)       else//计算新类与旧类之间的相似度，采用均值法
(14)         for m in range(len(nodes[i]))
(15)           for n in range(len(nodes[j]))
(16)             d=|LCS(nodes[i].m, nodes[j].n)|
(17)             sim_mn=2d/(|nodes[i].m|+|nodes[j].n|)
(18)             sum+= sim_mn
(19)         ava_sim_ij=sum/mn
(20)         if ava_sim_ij ≥ max_sim then
(21)           max_sim=ava_sim_ij
(22)           max_pair=(i, j)
(23)  if max_sim  ≥then //阈值判断
(24)    (i, j) = max_pair
(25)    node1 = nodes[i]
(26)    node2 = nodes[j]
(27)    del nodes[j]//删除nodes中要合并的特征，要求先删除j, j>i
(28)    del nodes[i]
(29)    nodes.append(node1.merge(node2, max_sim))//合并相似特征形成新簇，进入下一轮计算
(30)  else
(31)    return nodes[0]
(32)return nodes[0]
```

下面给出基于不一致系数的聚类树划分特征选择算法 SigSelect，如表 3 所示的算法 3。

算法 3 首先根据不一致系数的求解方法计算得到各个聚类链接处的不一致系数(第 1~7 行)，然后通过分析不一致系数和阈值的关系对聚类树进行划分(第 8~32 行)，最后从 M 分类集合中选择每一类的代表特征存入 SM 中(第 33~35 行)。其中算法 3 的 1~7 行的时间复杂度为 $O(m)$，第 8~32 行的时间复杂度也为 $O(m)$，第 33~35 行的时间复杂度为 $O(i)$，i 为划分的类数，$i \le m$，故算法 3 的时间复杂度仍为 $O(m)$。

表 3 特征选择算法

算法 3 SigSeclect

输入：聚类树 $Z[m-1][4]$，深度h，不一致系数增加幅度阈值 θ

输出：特征代表集SM[ ]

(1) **for** k=1 to $m-1$ do
(2) **if** ($Z[k][1] \ge 1$ && $Z[k][1] \le m$ && $Z[k][2] \ge 1$ && $Z[k][2] \le m$) then
(3) IC[k]=0 //叶子节点聚类处的不一致系数
(4) **else**
(5) $M(k)$=mean($Z(S_k,h)$)//第k次聚类处向下深度为h的所有链接长度的均值
(6) SD(k)=std($Z(S_k,h)$)//第k次聚类处向下深度为h的所有链接长度的标准差
(7) IC(k)=($Z[k][4]-M(k)$)/SD(k)//不一致系数
(8) tmp $=2m-1$
(9) **for** $k=m-2$ to 1 do
(10) **if** (IC(k+1) − IC(k))> θ then //根据不一致系数增值幅度进行划分
(11) x=$Z[k][1]$
(12) y=$Z[k][2]$
(13) **if** $x>y$ && tmp>x then
(14) $M[i]$=Z(tmp) − ($Z(x)Z(y)$)
(15) **if** $M[i]$!=NULL
(16) i++
(17) $M[i]$= $Z(x)$
(18) i++
(19) **else**
(20) $M[i]$=Z(tmp) − $Z(y)$
(21) i++
(22) tmp=y
(23) **if** $y>x$ tmp>y then
(24) $M[i]$=Z(tmp) − $Z(x)$
(25) **if** $M[i]$!=NULL
(26) i++
(27) $M[i]$=$Z(y)$
(28) i++
(29) **else**
(30) $M[i]$=Z(tmp) − $Z(x)$
(31) i++
(32) tmp=x
(33) **for** k=1 to i do //从划分的各特征子集中选择代表特征构成特征集
(34) **for** each S_j in $M[k]$
(35) SM[k]=selectlongest($M[k]$)
(36)return SM

选定特征之后，基于选定特征的内核恶意软件检测过程如下：

(1)对于待检测的软件样本，按照文献[11]的方法提取该样本的特征 S；

(2)根据基于最长公共子集的特征相似计算方法计算该特征与所选择特征的相似度，从中选择相似度最大的值 S_{sim}，若 $S_{\text{sim}} > \theta$，那么可判定该样本属于内核恶意软件，并给出其所属的类别；否则判定其为正常软件。

文献[11]中的检测方法时间复杂度为 $O(mn^2)$，而本文的检测方法时间复杂度为 $O(in)$，其中 m 为特征库样本特征数，n 为特征的长度，i 为划分的类数，$i \le m$。因此，本文的检测方法优于文献[11]的检测方法。

5 实验分析

本节从聚类分析有效性、特征选择有效性和性能 3 个方面对本文提出的方法进行评测。实验环境如下：主机 CPU 为 Intel(R) Core(TM) i5-750 @ 2.67 GHz，内存大小 4 GB，操作系统采用的是 3.2.43-x86_64 内核的 Ubuntu 12.04。选择了 620 种 linux 内核恶意软件样本作为测试用例，其中 610 种用于聚类分析，10 种用于验证特征选择的有效性。

5.1 聚类有效性

我们采用通用的聚类算法评价度量标准来验证算法 2 的有效性，一是准确率 P，二是召回率 R。准确率用于验证聚类算法区分不同样本的能力；召回率用于验证聚类算法识别相似样本的能力；值越大表明聚类结果与真实情况越相似，聚类效果越好。

令聚类相似度阈值为 0.7，那么算法 2 的准确率 P 为 0.985，召回率 R 为 0.942，两者取值均较高，这表明了算法 2 的有效性。

为了进一步验证选择最长公共子集进行数据特征聚类分析的合理性，我们实现了基于最长公共子串和最长公共子序列的特征聚类算法，这两者与算法 2 仅相似度计算方法不同，其余均一致。在相同测试样本和计算环境下，它们的准确率和查重率对比如图 2 所示。

从图 2 中我们可以看出，基于最长公共子集的聚类算法的准确率和召回率均是最高的。因此，选

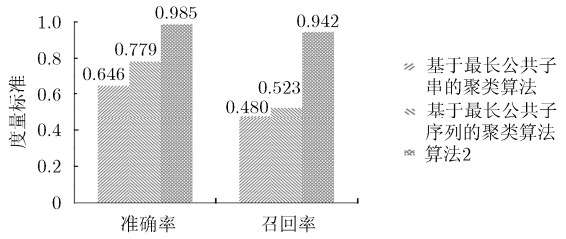

图 2 不同聚类算法的准确率和召回率

择最长公共子集计算相似度是合理的，它更适合于数据特征这种数据组成的相似度计算。

相似度阈值决定了两个样本是否是相似的。选择一个合适的阈值通常取决于预期的聚类粒度级别。例如，分析者可能想要得到一个较为粗的分类，那么阈值就可以设置得小一些；如果分析者想要得到一个细粒度的分类，那么阈值就需要设得大一些。

为了验证阈值选取的合理性，我们对不同阈值的聚类效果进行了统计分析，即对准确率和召回率进行分析。图 3 给出了准确率和召回率随着不同阈值的选择的变化趋势。从图 3 中可以看出，阈值范围为 0.6~0.9 可以得到较好的准确率和召回率。

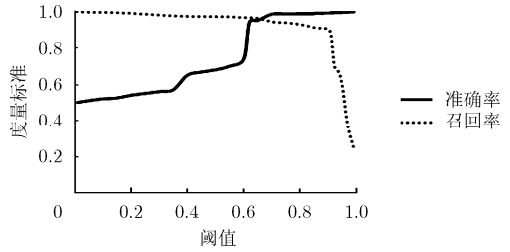

图 3 准确率和召回率与阈值关系

5.2 特征选择有效性

特征选择有效性分析主要用于验证选择的特征集能够有效区分不同的恶意软件。首先我们给出了基于特征选择算法得到的划分结果，然后我们选择了 10 款内核恶意软件来验证选择的特征的有效性。实验结果如表 4，表 5 所示。

从表 4 可知，测试样例经聚类划分为 14 类，每一类的个数和选择的恶意软件代表分别对应表 4 的第 2 列和第 3 列。

特征选择确定后，对于未知的样本，通过与每一类特征代表进行相似度计算判断该样本的归属。表 5 给出了待测的 10 个内核恶意软件的归属类别，与该类特征的相似度值，能否有效检测等信息。从表 5 可以看出，所有的恶意软件均归类到某一类别中，说明特征选择算法和检测方法是有效的。

表 4 恶意软件划分

类别编号	个数	选择的恶意软件
1	110	Suckit v2
2	130	Adore 0.38
3	15	wipemod
4	14	Linuxfu
5	16	JOP_hideprocess
6	110	Enyelkm v1.2
7	15	Mood-nt v2.0
8	110	Adore-ng 0.56
9	13	Dynamic_hook
10	15	All_root
11	15	Rkit
12	17	Suterusu
13	14	Rootkit-master
14	16	Disable_prng

表 5 特征选择有效性测试

恶意软件	类别	相似度	有效性
Adore_ng 1.56	8	0.930	√
Xingyiquan	2	0.760	√
Stmichael	2	0.824	√
Knark	12	0.736	√
Wnps	6	0.927	√
Disable_fireware	6	0.710	√
Fuuld	1	0.874	√
Get_root	10	1.000	√
Hideproc	5	0.880	√
Cleaner	3	1.000	√

5.3 性能测试

性能测试主要测试了聚类算法、特征选择算法在给定样本下的时间开销，并测试了随着阈值变化，聚类算法运行时间的变化情况，随样本变化，整体算法的运行时间开销，以及基于聚类分析的检测方法的时间开销。

(1)固定样本下的时间开销：　在阈值 $\theta = 0.7$ 时，该方法的运行时间为 6128 ms，各部分运行时间如表 6 所示。

表 6 各关键步骤运行时间(ms)

	一轮特征相似度计算	聚类算法时间	不一致系数计算时间	划分时间	特征选择时间	总时间
时间	28	5925	163	23	17	6128

(2)阈值与运行时间关系： 图 4 表明，阈值在达到 0.6 前，聚类算法运行时间波动不大。当阈值大于等于该值后，聚类算法运行时间大幅度降低，并且在阈值达到 0.9 之后，聚类算法运行时间趋于平稳。这也与准确率和召回率的变化情况相吻合，当阈值小于 0.6 时，样本之间进行合并的几率较大，合并次数较多，故需要较多的时间；当阈值大于 0.9 时，样本进行合并的几率已经很小，故算法所需时间变化不大，逐步趋于平稳。

(3)样本与运行时间关系： 在聚类算法阈值 $\theta = 0.7$ 时，通过不同的样本测试算法的时间开销，实验结果如图 5 所示。

从图 5 可知，随着样本的增加，聚类算法所需的时间增加幅度较大，呈指数级增长，与上文分析的时间复杂度一致，特征算法选择时间变化较小，呈线性变化趋势，与上文分析的时间复杂度也一致。尽管在聚类时花费了较多时间，但这将为恶意软件检测提供较高的检测效率。

(4)检测方法时间开销对比： 通过对不同待检测样本情况下的检测时间开销统计(不包括特征提取时间开销)，对比检测方法的时间开销。实验结果如图 6 所示。

根据图 6，本文的检测方法时间开销远小于文献[11]的方法，这与时间复杂度分析是相符合的。在样本特征长度 n 一致的情况下，文献[11]的检测方法时间复杂度 $O(mn^2)$ 可视为 $O(m)$，本文的检测方法时间复杂度 $O(in)$ 可视为 $O(i), i \le m, m$ 为样本特征数，它们均与样本数呈线性关系，差异在于系数不一样。

6 结束语

本文深入研究了内核恶意软件检测方法，尤其是基于数据特征的检测方法。数据特征描述了恶意软件对内核对象的篡改操作，能够较好地适用于检测内核恶意软件。但由于随着样本特征的增多，基于数据特征的检测方法效率随之降低，故研究了基于聚类分析的特征选择方法。该方法首先分析了现有特征相似性计算方法用于数据特征计算时存在的不足，提出了最长公共子集算法，然后讨论了基于最长公共子集的特征相似性计算和特征选择算法，这使得该方法可快速地对未知样本进行分析和分类，从而提高恶意软件检测效率。最后通过实验验证了该方法的有效性。下一步工作研究将基于最长公共子集的相似度计算应用于其它聚类算法的有效性和性能。

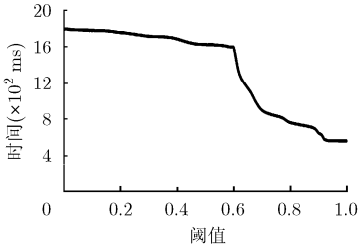

图 4 阈值变化与算法运行时间关系

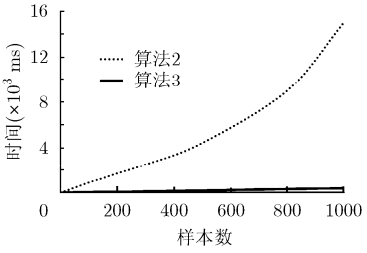

图 5 算法运行时间与样本变化关系

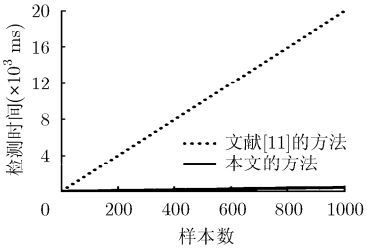

图 6 检测方法时间开销

参 考 文 献

[1] Yin H, Song D, Egele M, *et al*.. Panorama: capturing system-wide information flow for malware detection and analysis[C]. Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, USA, 2007: 116–127.

[2] 王蕊, 冯登国, 杨轶, 等. 基于语义的恶意代码行为特征提取及检测方法[J]. 软件学报, 2012, 23(2): 378–393.
Wang Rui, Feng Deng-guo, Yang Yi, *et al*.. Semantics-based malware behavior signature extraction and detection method[J]. *Journal of Software*, 2012, 23(2): 378–393.

[3] Nataraj L, Karthikeyan S, Jacob G, *et al*.. Malware images: visualization and automatic classification[C]. Proceedings of the 8th International Symposium on Visualization for Cyber Security, Pittsburg, PA, USA, 2011: 4–10.

[4] Nataraj L, Yegneswaran V, Porras P, *et al*.. A comparative assessment of malware classification using binary texture analysis and dynamic analysis[C]. Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, Chicago, USA, 2011: 21–30.

[5] 韩晓光, 曲武, 姚宣霞, 等. 基于纹理指纹的恶意代码变种检测方法研究[J]. 通信学报, 2014, 35(8): 125–136.
Han Xiao-guang, Qu Wu, Yao Xuan-xia, *et al*.. Research on malicious code variants detection based on texture fingerprint [J]. *Journal of Communications*, 2014, 35(8): 125–136.

[6] Ding Yun-xin, Dai Wei, Yan Sheng-li, *et al*.. Control flow-based opcode behavior analysis for malware detection[J]. *Computer & Security*, 2014, 44: 65–74.

[7] Wang X and Karri R. NumChecker: detecting kernel control-flow modifying rootkits by using hardware performance counters[C]. Proceedings of the 50th Annual Design

Automation Conference, Austin, TX, USA, 2013: 79–86.

[8] Debbabi M, Desharnais J, *et al*.. Static detection of malicious code in executable programs[J]. *International Journal of Requirement Engineering*, 2001(184–189): 79–86.

[9] Baliga A, Ganapathy V, and Iftode L. Detecting kernel-level rootkits using data structure invariants[J]. *IEEE Transactions on Dependable and Secure Computing*, 2011, 8(5): 670–684.

[10] Zhu F. Integrity-based kernel malware detection[D]. [Ph.D. dissertation], Florida International University, 2014.

[11] Rhee J, Riley R, Lin Z Q, *et al*.. Data-centric OS kernel malware characterization[J]. *IEEE Transactions on Information Forensics and Security*, 2014, 9(1): 72–87.

[12] Tumer D, Entwisle S, Fossi M, *et al*.. Symantec Internet security thread report 2014[R]. Symantec Corporation, 2014.

[13] 陈季梦, 陈佳俊, 刘杰, 等. 基于结构相似度的大规模社交网络聚类算法[J]. 电子与信息学报, 2015, 37(2): 449–454.
Chen Ji-meng, Chen Jia-jun, Liu Jie, *et al*.. Clustering algorithms for large-scale social networks based on structural similarity[J]. *Journal of Electronics & Information Technology*, 2015, 37(2): 449–454.

[14] Ciprian O, George C, and Gheorghe S. Malware clustering using suffix trees[J]. *Journal of Computer Virology Hacking Techniques*, 2014, DOI: 10.1007/s11416-014–0227-6.

[15] 戚树慧. 基于指令分析的恶意代码分类与检测研究[D]. [硕士论文], 杭州电子科技大学, 2012.
Qi Shu-hui. Research into malware classification and detection based on instruction analysis[D]. [Master dissertation], Hangzhou Dianzi University, 2012.

[16] 罗养霞, 房鼎益. 基于聚类分析的软件胎记特征选择[J]. 电子学报, 2013, 41(12): 2334–2338.
Luo Yang-xia and Fang Ding-yi. Feature selection for software birthmark based on cluster analysis[J]. *Acta Electronica Sinica*, 2013, 41(12): 2334–2338.

[17] Bailey M, Oberheide J, Andersen J, *et al*.. Automated classification and analysis of internet malware[C]. Proceedings of the 10th Symposium on Recent Advances in Intrusion Detection, Gold Coast, Australia, 2007: 178–197.

陈志锋： 男，1986 年生，博士生，研究方向为信息安全与可信计算.

李清宝： 男，1967 年生，教授，研究方向为信息安全与可信计算.

张　平： 女，1969 年生，副教授，研究方向为并行识别、信息安全.

冯培钧： 男，1990 年生，博士生，研究方向为信息安全.