doi: 10.3969/j.issn.0490-6756.2018.03.009

基于模糊哈希特征表示的恶意软件聚类方法

肖锦琦，王俊峰
(四川大学计算机学院，成都 610065)

摘　要：目前，每年被拦截到的新型恶意软件变种数已达千万级别，在线恶意软件仓库 Virus Share 上存储的未分类的恶意软件数量也超过了 2700 万. 将恶意软件按一定的行为模式进行聚类，不仅使新型攻击更易被检测出来，也有助于及时获取恶意软件的发展态势并做出防范措施. 因此提出了一种高效的恶意软件聚类方法，对恶意样本进行动态分析并筛选出包括导入、导出函数、软件字符串、运行时资源访问记录以及系统 API 调用序列等特征，然后将这些特征转换为模糊哈希，选用 CFSFDP 聚类算法对恶意软件样本进行聚类. 并将聚类个数、准确率、召回率、调和平均值以及熵作为聚类效果的外部评估指标，将簇内紧密度以及簇间区分度作为内部评估指标，实验结果表明，与 Symantec 和 ESET-NOD32 的分类结果相比，本文提出的方法的聚类家族个数与人工标记的数量最为接近，调和平均值分别提升 11.632%，2.41%.

关键词：恶意软件家族；聚类；模糊哈希；特征提取
中图分类号：TP309.7　　**文献标识码**：A　　**文章编号**：0490-6756(2018)03-0469-08

A malware variant clustering method based on fuzzy hash

XIAO Jin-Qi, *WANG Jun-Feng*
(College of Computer Science, Sichuan University, Chengdu 610065, China)

Abstract: Internet Security companies collect tens of millions of new malware variants each year, Virus Share, the online malware repository, has stored more than 27 million unlabeled malwares. Clustering malware variant according to certain behavior patterns, not only makes the new attack easier to be detected, but also helps us to obtain the malware trends in time and take the corresponding preventive measures. Therefore, this paper proposes a malware variant clustering method which use dynamic analysis technology to extract malware features, including import and export function name, strings, system resource records and system calls, then convert these features to the fuzzy hashes, finally clustering malware samples through the CFSFDP clustering algorithm. We select the number of clusters, precision, recall, F-score and entropy as external criteria, select the intra-cluster cohesion and inter-cluster separation as internal criteria. The experimental results demonstrate that compared with Symantec and ESET-NOD32, the F-score obtained in this paper increased by 11.632% and 2.41%, and the number of clusters is closest to the artificial labeled.

Keywords: Malware family; Clustering; Fuzzy Hash; Extract features

收稿日期：2017-08-11
基金项目：国家重点研究与发展项目(2016YFB0800605)；国家自然科学基金(91338107，91438119，91438120)；国家教育部博士点基金(20130181110095)；四川省重点科学技术研究发展项目(2016ZR0087)
作者简介：肖锦琦(1992－)，男，硕士生，研究方向为网络与信息安全. E-mail:308574223@qq.com
通讯作者：王俊峰. E-mail:wangjf@scu.edu.cn

1 引 言

网络犯罪和网络恐怖主义对社会造成的危害日益严重,由于该领域的复杂性、全球性以及动态性,我们所采取的相应政策和措施仍然难以完全抵抗其威胁[1]. 在 CAPITAL(Cybersecurity Research Agenda for Privacy and Technology Challenges)项目所列出的互联网威胁列表中,恶意软件是存在最久且最为严重的威胁之一,它们在未经用户许可的情况下执行恶意操作(如锁定浏览器首页,窃取或破坏用户资料,诱导用户进行某些操作等)以获取不正当利益[2]. 赛门铁克公司在2016年的互联网安全威胁报告中提到,其在2015年共检测到4.31亿个新型变种,平均每天发现的恶意变种数量已达到百万级别,而在2014年检测到的新型变种数量则为3.17亿. 另外,据 Check Point 的统计,在2015年检测到的僵尸网络攻击中,Sality、Conficker、ZeroAccess 和 Cutwail 这四种类型占了总数的50%且都来源于已知的恶意软件. 实际上每年新增的恶意软件家族数量呈下降趋势,大多数恶意软件经过混淆或修改小部分代码后就可被重复利用. 恶意软件这种低成本、大批量的生产模式导致基于哈希签名的检测方式存在较大的滞后性,同时也为恶意软件的分析带来了巨大的挑战. 因此,采取自动化、高效率且细粒度的分析方式十分必要,在获取恶意软件的行为信息后,按照家族特征进行聚类,这一过程过滤了常见的恶意软件攻击方式,新型的攻击行为或者躲避检测的行为都会以新的类型或者噪声的形式被提取出来,这对实时获取威胁情报以及互联网安全态势都很有意义.

2 相关工作

恶意软件聚类是指通过自动化的手段,根据恶意软件的行为、目的以及外在特征等特点自动将最为相似的恶意软件聚集在一起. 目前针对恶意软件聚类的研究主要包含以下四个方面:恶意软件的特征提取、特征表示、聚类算法的选择及优化、聚类效果评估.

静态分析方面,Yanfang 等人通反汇编恶意软件程序提取样本指令序列及频度,集成 TF-IDF 及 k-medoids 的聚类方法实现分类[3]. 但由于混淆技术的存在,反汇编技术难以从中提取出完整的指令序列,且由于采用 TF-IDF 进行统计,也在一定程度上削弱了聚类算法无需事先训练的特点. Mansour Ahmadi 等人[3]从二进制文件以及反汇编文件中提取静态特征,包括字节码 N-gram、信息熵、字符串信息、二进制信息图、操作码特征、寄存器使用频率、Windows API 接口、各节区的特征等. 通过多维度的特征信息,该方法将9个恶意软件家族正确分类,正确率为99.77%. 但恶意软件家族远不止9种,且该方法采用的是有监督的学习,样本的覆盖率决定了最后的准确率. 诸如加壳、修改 PE 格式、抗反汇编、反调试等混淆技术手段也使得提取出静态特征很难保证持续有效性.

动态分析方面,Cesare 等人利用信息熵检测恶意软件是否加壳,并对加壳软件进行脱壳,然后从生成的汇编代码中提取控制流程图作为样本特征,再通过近似图匹配算法实现恶意代码的分类[4],但该方法的距离范围是0~1.00,且该方法主要是用来区分恶意与良性样本,在样本和家族种类数量过多时难以保证个家族间有足够的区分度. Bojan Kolosnjaji[5]通过 Cuckoo Sandbox 提取样本系统 API 调用信息,通过卷积神经网络和递归神经网络层建立系统 API 调用模型,其中,卷积神经网络使用基于 API 的 n-gram 特征序列,递归神经网络训练则使用全序列特征. 实验结果表明使用该方法的分类效果要好于单独使用神经网络、隐马尔科夫模型和 SVM 的方法,但神经网络方法需要大量数据训练和参数的调优且目前并没有足够有效的人工标记样本用来训练,提升了该方法实际应用的难度.

在特征表示方面,目前大都采用基于统计的特征表示方法,如文献[6]等. 特征进行统计得到特征个数以及每个特征的频度等信息,然后生成维数固定的特征向量,这使得后续的分类工作极大地依赖于前期样本的选择. 如文献[7]通过建立行为依赖链来表示特征,首先提取了160种 API 并将其分为6中行为类型,每个行为由一个6元组表示,并对每种行为做了统计,得到相应的权重值最后通过对比两个链表的交集与并集计算出 Jaccard 距离,得到相似度.

综合来看,以上方法从不同层面选取了恶意软件的特征,通过分类或者聚类方法实现了恶意软件家族的判别,但仍存在一些问题:(1) 由于混淆技术以及反调试技术的出现,基于静态分析的方法难以提取到有效的特征信息;(2) 特征的表示大都需要先进行全局的统计以保证有足够的维度来表示特征,这导致提取的特征过于依赖现有的样本,使得生成的模型难以

具有普适性；(3) 使用有监督的训练方式对样本的覆盖率有严格要求，需要事先人工标注样本进行学习，学习的家族数决定了识别的家族数，而恶意软件家族数量庞大难以逐一标记；(4) 在聚类算法的选择上，仅通过与事先标记好的样本作对比难以有足够的说服力，一方面家族种类的定义与标记的方法没有严格准确的说明，另一方面没有考虑到样本标记出错时对实验结果带来的影响.

针对以上问题，本文对恶意软件样本的静态特征、动态特征(API 调用序列以及资源操作记录)做了深入分析，将这些特征重新筛选、组织后转换为模糊哈希，通过模糊哈希的距离度量公式来计算两个样本间的距离. 为了获取较为理想的聚类算法，本文选取了 4 种聚类算法并采用多维的评价指标进行对比. 实验结果显示，与现有的 12 种主流杀毒软件相比，本文提出的方法产生的聚类效果更好.

3　系统设计

本文所提方法主要包括特征提取、特征表示以及聚类三个主要模块，整体的框架如图 1 所示.

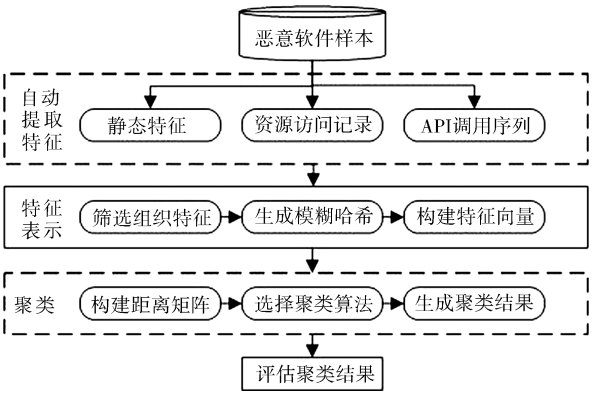

图 1　系统整体架构

Fig. 1　Architecture description

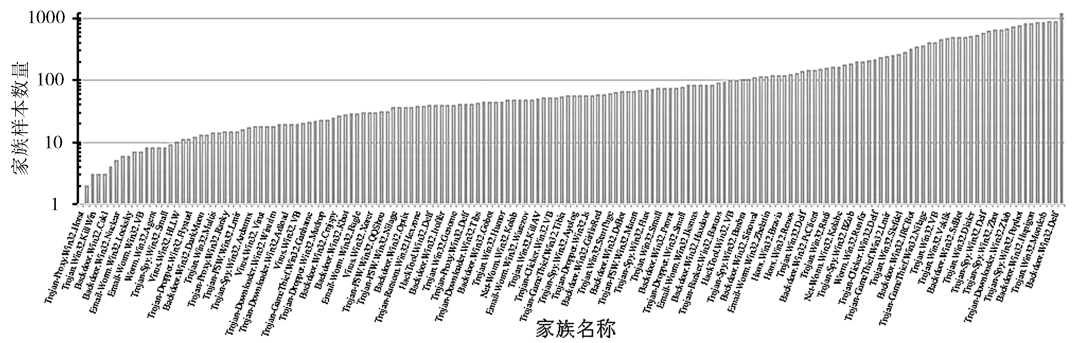

图 2　恶意软件家族样本分布

Fig. 2　The distribution map of malware samples

3.1　静态特征的选取

在本文采用的恶意软件样本来自 VX-Heaven，包含了 24855 个样本，164 个家族，通过 PEID 检测到的加壳率为 30.66%，平均每个家族变种大约用到 6 种不同的加壳工具，其家族样本分布如图 2 所示.

对于 PE(Portable Executable)文件来说，其静态特征主要包括 PE 头部数据、节区数据、反汇编代码等. 恶意软件为了便于传播，大多数会通过压缩壳、或者调用动态链接库(Dynamic Link Library)的方式来减小自身体积，并利用反调试、抗反汇编、反虚拟机等技术避免查杀. 如此多的混淆手段使得从样本中提取的字节码序列、汇编指令序列、熵值、节区信息等静态特征极易被修改，这些信息对于判别恶意或良性软件或许有较好的结果，但仍难以有效的对恶意软件中各个家族进行区分. 因此，在静态特征中，应提取那些不易被修改或者难以伪装的信息以保证特征的长期有效性.

Windows 操作系统使用 PE 导入表实现 PE 文件动态链接库的加载及重定位，PE 导入表包含 PE 文件调用外部函数名称、以及动态链接库的信息. 而 PE 导出表描述了其所在的 PE 文件中向其他程序提供的可供调用的函数的情况. 而在二进制文件中所提取的字符串，不仅可能存在文件的版本信息，也可能存在一些运行过程中所包含的提示信息、函数信息等. 如果恶意软件通过混淆的方式隐藏了这些信息，那在动态分析的过程中，这些信息仍有可能暴露出来. 因此本文提取了样本 PE 文件头中的导入和导出函数(Import/export functions)信息、字符串信息作为选取的静态特征.

3.2　动态特征的选取

混淆技术使得恶意软件隐匿了自身的外在特征，通常的静态分析技术难以挖掘到更深层次的信息. 动态分析技术通过在虚拟环境中运行恶意软件

来记录其API调用顺序以及对文件、注册表、网络等系统资源的操作.本文将动态特征分为API调用序列和运行时资源访问记录两类,分别用于表示软件在启动、运行过程中为完成某一行为所需要的系统函数以及对系统资源造成的影响.

(1) API调用序列.

API调用序列是恶意软件行为的一种外在表现方式,但仅提取函数名和参数并不能有效地体现其真实的行为特征,例如ShellExecute(),其最有价值的信息是参数中的shell命令,LdrLoadDll()里面存在大量参数,但最有用的是其要加载的DLL名称,而对于LoadString()、DrawText()等函数会在调用序列中大量存在实际上干扰了恶意软件的真实意图等等.因此,本文将API的类型分为:system、process、ui、misc、network、__notification__、resource等,并对每种类型的特征进行分析过滤以获取更有价值的信息.

system类型记录在Windows内核层级的调用,而负责操作系统基本功能(如I/O、内存管理、进程和线程创建等)的KERNEL32.DLL,负责注册表相关操作的ADVAPI32.DLL以及负责网络通信的WS2_32.DLL等windows下的动态链接库,通过内核态的LdrLoadDll()或LdrGetDllHandle()函数进行加载,通过LdrGetProcedureAddress()调用相关函数.因此在调用以上三个函数时,不再记录函数名称而记录其所加载的DLL以及DLL所调用的函数名称.

process类型记录了对内存、进程执行操作的函数,如内存的映射、分配与释放、Shell命令的执行等.其中,执行Shell命令的函数是ShellExecuteEx(),它可以指定某个程序或文件执行一些操作,因此需要记录的是执行的命令以及执行该命令的进程名称.

ui类型记录了对窗口、控件的外观资源进行操作的函数,对查找制定窗口的函数FindWindowA()提取其加载的窗口的类名、对加载字符串的函数LoadString()、DrawText()提取字符串信息.在对恶意软件的分析中发现,这些类型的操作会存在大量的重复,如查找某个指定窗口或弹出某些特定信息等.因此单独建立一个树状图,按字典升序排序,并剔除重复项,并将这些信息排在整个类型特征的最后,以避免对样本实际执行流程记录的干扰.

misc类型记录了一些杂项,函数GetSystemMetrics()可获取系统数据或配置信息,这里仅提取其要获取的信息类型;函数WriteConsoleW()可从当前光标位置写入字符串到屏幕缓冲区,这里仅提取其写入的内容.其余的诸如获取鼠标位置、创建激活上下文环境等函数只提取其函数名.

network类型负责记录与网络通信相关的函数,在提取函数名称时,如果参数有URL(Uniform Resource Locator)、IP地址或者域名信息时,应提取出来附加在函数名称后.

__notification__和resource类型分别是记录异常信息和资源加载的函数操作,并不属于程序执行流程的主体,因此不提取这两类的函数名称.其他不属于以上特定类型的函数,仅提取其函数名称.

(2) 运行时资源访问记录.

运行时资源访问记录反映了恶意软件运行时对操作系统产生的影响.本文将运行时的资源访问记录分为文件、注册表、DLL、互斥量(Mutex)、GUID(Globally Unique Identifier)以及其他等六种类型,相关描述见表1.

表1 资源访问记录

Tab.1 The description of resource access records

资源类型	描述
文件	样本在运行过程中创建、删除、读取以及修改的文件路径
注册表	样本在运行过程中创建、删除、读取以及修改的注册表路径
DLL	样本所加载的动态链接库名称
互斥量	样本所使用的互斥量名称
GUID	GUID的十六进制数字标识符
其他	包括要解析的域名、执行的DOS命令等

3.3 基于模糊哈希的特征表示方法

由于恶意软件变种增速迅速、家族数量庞大,基于统计的特征表示方法要保持自身的长期有效性,就必须不断地更新特征空间扩充特征向量的维度.而模糊哈希算法又称为相似性摘要算法(Similarity digest)[8,9],主要用于文件的相似性比较.由于从恶意软件中提取的特征大都是文本特征,诸如API调用序列,PE格式信息、注册表信息、字符串信息等,因此使用模糊哈希表示这些特征时较为合适的.将特征转换成模糊哈希后通过自身特定的距离度量函数度量两个哈希值之间的距离.使用模糊哈希进行特征表示与基于统计的特征表示方法相比有如下四个优点:(1) 无需事先统计特征便可表

示任意的属性值；(2) 将大量的属性特征压缩成为一组哈希值，极大地降低了特征向量的维度；(3) 其距离(相似度)度量函数是量身定做的，无需再去寻找合适的距离度量函数；(4) 对文本变化的度量精度要优于基于统计的特征表示方法. 模糊哈希的计算方法[9]的步骤如下：

1) 用大小为 5 个字节的滑动窗口处理目标字符串 S，一次向前滑动一个字节，设一个滑动窗口的内容为：ABCDE；则分别采用 Pearson Hash 映射并统计 ABC、ABD、ABE、ACD、ACE、ADE 这 6 个 bucket 的个数；

2) 定义 q_1、q_2、q_3 为：75%的 bucket 的个数$\geqslant q_1$，50%的 bucket 的个数$\geqslant q_2$，25%的 bucket 的个数$\geqslant q_3$；构造模糊哈希的头部，共三个字节：第一个字节是字符串的校验和；第二个字节表示字符串 S 的长度 L；第三个字节由两个 16 位的数 q_{1_ratio}、q_{2_ratio} 组合而成，计算方法如式(1)所示.

$$\begin{cases} q_{1\_ratio} = (q_1 * 100/q_3) \bmod 16 \\ q_{2\_ratio} = (q_2 * 100/q_3) \bmod 16 \end{cases} \tag{1}$$

3) 构造模糊哈希的主体部分：Pearson Hash 生成的 128 个映射中每个映射为一个键，其对应的值为该键出现的个数；遍历每个键的值 $value$，当 $value \leqslant q_1$ 时，输出“00”；当 $q_1 < value \leqslant q_2$ 时，输出“01”；当 $q_2 < value \leqslant q_3$ 时，输出“10”；当 $value > q_3$ 时，输出“11”；最终得到大小为 32 字节的主体部分哈希值：

4) 将步骤 2)和 3)所得二进制串转换为 70 个十六进制数，得到最终的模糊哈希值. 设两个模糊哈希值为 X 和 Y，定义 $\mathrm{mod\_}diff(a,b,R)$ 为一个在大小为 R 的循环队列中 a 到 b 的最小距离，即

$$\mathrm{mod\_}diff(a,b,R) = \mathrm{Min}((a-b) \bmod R, (b-a) \bmod R) \tag{2}$$

5) 使用 mod_$diff$() 分别计算两个模糊哈希值 X 和 Y 头部属性中 L、q_{1_ratio} 和 q_{2_ratio} 的值，其中 $R=256$，若所得结果值 $diff>1$，则令 $diff=(diff-1)*12$，否则 $diff$ 保持不变；然后计算两个 TLSH 值 X 和 Y 的校验和的距离，如果 X 和 Y 校验和相等则校验和的距离为 0，否则为 1；以上所求结果相加即为头部之间的距离值.

6) 计算 X 和 Y 主体间的距离：将 X 和 Y 主体所占的 256 个二进制位分别从左往右平分为 128 组，每组的二进制由 $b_0\,b_1$ 表示；如果 X 与 Y 某组应用的二进制值不相等则按照式(3)进行计算，否则该组的距离为 0，然后将 X 与 Y 每组对应的距离累加起来即为主体间的距离.

$$(X_{b_0 b_1} Y_{b_0 b_1}) = \begin{cases} 1100 \text{ 或 } 0011, & \text{距离为 } 6 \\ 1101 \text{ 或 } 0010, & \text{距离为 } 2 \\ \text{其他}, & \text{距离为 } 1 \end{cases} \tag{3}$$

7) 将 X 和 Y 头部间的距离与主体间的距离相加即为两个模糊哈希值 X 和 Y 之间的距离值.

本文选用静态特征对基于统计的特征表示方法与基于模糊哈希特征表示方法进行对比. 一方面，将这些特征提取排序后直接转换为模糊哈希值作为特征信息进行聚类；另一方面，选用了两种基于统计的方法表示特征，在 24855 个样本中共得到了 45016 个特征，第一种通过计算其 TF-IDF (Term Frequency-Inverse Document Frequency) 值得到权重，并使用余弦距离以及欧氏距离作为距离度量方法；第二种不计算权重值仅通过二进制表示特征是否存在，采用 Jaccard 距离作为距离度量方法. 用 K-Medoids 对这三种不同的特征表示方法进行聚类后的结果如表 2 所示，其中加粗显示的数据为该列的最优值.

表 2　不同特征表示方法聚类效果对比

Tab. 2　The comparison of feature expression methods

特征表示方法	聚类个数	准确率	召回率	F 值	熵值
模糊哈希	177	0.257	0.146	0.147	3.507
TF-IDF(余弦距离)	4	0.020	0.559	0.037	0.435
TF-IDF(欧氏距离)	5	0.020	0.559	0.037	0.435
二进制(Jaccard 距离)	4	0.020	0.559	0.037	0.435

由此可以看出，使用模糊哈希进行特征表示要比基于统计方法得到的特征向量具有更好的区分度，除了特征表示的问题外，距离度量公式也是影响聚类结果的重要因素. 余弦距离仅度量了两个样本间的夹角但没有考虑到样本间的绝对距离，而欧氏距离则只计算了两个样本间的绝对距离，Jaccard 距离则过于笼统，仅考虑到了特征的有无. 另外基于统计方法得到的特征是在现有样本的基础上筛选得到的，权重的大小也取决于样本的分布，而基于模糊哈希的特征表示方法则无需考虑这些.

4　恶意软件聚类

4.1　聚类效果评估标准

对聚类效果的评价又称为聚类有效性分析[10] (Cluster Validity)，主要分为三类：第一类称外部

评估指标(External Criteria),用事先判定的聚类结构来评价聚类结果;第二类是内部评估指标(Internal Criteria),用参与聚类的样本来评价聚类结果,比如采用聚类结果中各个簇的误差平方和;第三类是相对标准(Relative Criteria),用同一算法不同参数得到的结果来评价,通过与其他结果的比较来判断聚类结果的优劣.

(1) 外部评估指标.

外部评估指标包括:准确率、召回率、调和平均值、噪声个数、簇的个数以及熵值.其中准确率侧重于评价簇本身的凝聚程度;召回率侧重评价聚类结果与人工标记的整体吻合程度;调和平均值则是准确率与召回率的综合考量;噪声是指在基于密度的聚类算法中由于样本所在区域密度较小未达到生成簇的最低要求的那些样本;簇的个数可表示检测到的恶意软件家族数量;熵值代表了聚类结果的混杂程度.各指标的计算公式如下.

设样本集合为 S,则对于任意一个样本 t,$\forall t\in S$,可定义准确率(P)、召回率(R)、调和平均值(F)、熵($entropy$)分别由式(4)~(7)表示.

$$P(t)=\frac{|L_t\cap C_t|}{|C_t|},\ P=\frac{1}{n}\sum_{i=1}^{n}P(t_i) \tag{4}$$

$$R(t)=\frac{|L_t\cap C_t|}{|L_t|},\ R=\frac{1}{n}\sum_{i=1}^{n}R(t_i) \tag{5}$$

$$F(t_i)=\frac{2\times P(t_i)\cdot R(t_i)}{P(t_i)+R(t_i)},\ F=\frac{1}{n}\sum_{i=1}^{n}F(t_i) \tag{6}$$

其中,L_t表示人工标记结果中包含样本 t 的簇的集合;C_t表示聚类结果中包含样本 t 的簇的集合;n 表示样本总数;t_i表示第 i 个样本.

$$Pr_i(C_j)=\frac{|L_i\cap C_j|}{|L_i|}entropy(L_i)=-\sum_{j=1}^{k}Pr_i(C_j)\cdot\log_2 Pr_i(C_j)$$

$$entropy=\sum_{i=1}^{m}\frac{|L_i|}{|L|}\cdot entropy(L_i) \tag{7}$$

其中,L_i表示人工标记的第 i 个簇的集合;C_j表示聚类生成的第 j 个簇的集合;$|L_i|$表示集合L_i包含的样本个数;k 为人工标记的簇的个数;m 为聚类生成的簇的个数.

(2)内部评估指标.

本文选取的内部评估指标包括簇内样本数、簇内紧密度(Intra-cluster Cohesion),簇间区分度(Inter-cluster Separation).设 noc 为簇的个数,$L_i(j)$为簇L_i内的第 j 个样本,$L_i(center_j)$表示该簇内第 j 个核心点,$|L_i(center)|$表示簇L_i内核心点的个数,两个样本 x,y 间的距离为 $distance(x,y)$,簇内紧密度和簇间区分度的计算方法如式(8)和(9)所示.

$$\text{簇内紧密度}=\sum_{i=1}^{noc}\frac{\sum_{j=2}^{|L_i|}\sum_{k=1}^{|L_i|}distance(L_i(j),L_i(k))}{|L_i|\cdot(|L_i|-1)/2}(j>k) \tag{8}$$

$$\text{簇间区分度}=\text{Min}\left(\frac{\sum_{m=1}^{|L_i(center)|}\sum_{n=1}^{|L_j(center)|}distance(L_i(center_m),L_j(center_n))}{m\cdot n}\right)(noc\geq i>j\geq 1) \tag{9}$$

其中,簇内紧密度是通过计算簇内所有样本之间距离的平均值得到,结果越小说明该簇内的样本相似度高,凝聚程度较好.每个簇内有一个或多个核心点,簇与簇之间的距离为两个簇内核心点距离的平均值,取簇间距离最小的值为簇间区分度,簇间区分度越高说明簇与簇之间相似度低,离散程度好.一个聚类结果中,如果簇内紧密度小,簇间区分度高,则说明该聚类结果较优.

4.2 聚类算法对比与选择

聚类是一个将数据集中相似的数据成员进行分类组织的过程,既可以用于寻找数据内在的分布结构,也可以作为分类等其他学习任务的前驱过程.目前还没有一种聚类算法可较好的适用于揭示不同数据集所呈现出来的多样的结构[11].本文使用的恶意软件样本分布如图 2 所示,由于提取的特征为文本类型,恶意软件家族的分布也不会呈现规则的几何形状,因此本文选取能够处理混合数据类型或能够发现任意形状簇的 DBSCAN、OPTICS 以及 Clustering by fast search and find of density peaks(CFSFDP)[12]三种基于密度的聚类算法进行对比,其中特征的表示与距离度量方法均采用模糊哈希方法进行.同时选用 12 种杀毒软件检测的结果作为参考,结果如表 3 所示.

表 3　外部评估指标对比

Tab. 3　The comparison of external Criteria

检测方法	噪声个数	簇的个数	准确率(%)	召回率(%)	调和平均值(%)	熵值
McAfee	19	9385	75.065	13.936	17.079	3.791
Symantec	101	512	32.465	27.924	21.876	2.791
ESET-NOD32	82	2077	59.210	31.455	31.098	3.284
TrendMicro	60	5996	59.357	14.869	13.797	6.046
Avast	59	3504	53.090	18.247	15.069	3.712
ClamAV	55	8994	56.738	23.060	6.114	1.840
BitDefender	7	7927	71.188	12.413	15.250	4.435
DrWeb	114	7410	71.262	15.748	17.836	3.398
Microsoft	30	2108	60.304	23.614	26.180	4.202
AVG	49	9708	74.528	10.357	12.827	3.418
Panda	225	1596	29.419	26.221	12.982	1.589
Qihoo-360	2351	2369	23.676	15.244	6.136	7.261
OPTICS-300-33	822	96	32.345	55.455	28.836	0.941
OPTICS-100-8	3923	169	35.117	42.086	25.021	0.893
DBSCAN-70-9	942	184	44.615	46.318	32.263	1.219
CFSFDP-120-5	2178	162	44.666	45.219	33.508	1.225

可以看到，在杀毒软件中，ESET-NOD32 与 Symantec 的检测结果较好，但是生成的簇数较多与标准偏差太大，熵值较高. 虽然大多数杀毒软件准确率较高，但是簇的个数偏多，说明很多样本实际上并没被聚集在一起，即一个家族可能只包含了一个或几个样本，这导致了准确率偏高但召回率却偏低的现象. 而除了噪声个数和准确率外，其他最优值均出现在基于模糊哈希特征表示的聚类算法中，CFSFDP 算法得到了最高的调和平均值且得到了较为均衡的结果，与 Symantec 和 ESET-NOD32 的分类结果相比调和平均值分别提升 11.632%，2.41%，且生成的簇数与标准(164 个)较为接近.

表 4　内部评估指标对比

Tab. 4　The comparison of Internal Criteria

检测方法	噪声	最大簇内个数	簇内紧密度	簇间区分度	最大簇内紧密度
DBSCAN(70,9)	942	5579	66.28	169.87	181.80
OPTICS(100,8)	3923	6619	51.90	131.80	224.73
OPTICS(300,33)	822	8698	98.69	196.32	274.79
CFSFDP(120,5)	2178	4454	80.74	183.06	109.56

表 4 为内部评估指标对比，其中基于密度的聚类算法(CFSFDP、DBSCAN 和 OPTICS)生成的簇较为紧密，且簇与簇之间的区分度更高；其中，CFSFDP 最大簇的紧密度较好，而 OPTICS 与 DBSCAN 则在最大簇内的紧密度相对松散，簇的质量相对较差，整体来看 CFSFDP 算法得到了较为均衡的结果.

对比几种聚类算法，由于 DBSCAN 和 OPTICS 对参数较为敏感只能依靠经验选择，而 CFSFDP 可以通过生成决策图来对参数进行选择，且聚类结果也相对优异，因此使用 CFSFDP 算法进行聚类是较好的选择.

4.3　CFSFDP 算法参数的选择

CFSFDP 算法所需要输入的参数有截断距离 d_c、局部密度 ρ 以及距离 δ. 其中 d_c 需要事先确定，文献[20]指出在基于若干数据集的测试后发现当 d_c 使得每个数据点的平均邻居个数为数据点总数的 1%～2%时效果最佳，因此，在计算完样本各点之间的距离后，可通过这一点来确定 d_c 的值. ρ_i 表示与数据点 i 间距离小于 d_c 的数据点的个数；δ_i 表示当数据点 i 具有最大局部密度时，与其它数据点间最远的距离；否则 δ_i 表示在所有局部密度大于 ρ_i 的数据点中，距离数据点 i 最近的距离. 至此，对于每个数据点都可以算得 (ρ_i,δ_i)，并由此生成一个决策图. 本文采用的样本所得到的决策图如图 3 所示.

对于一个聚类中心，应同时具有较大的 ρ_i 和 δ_i，也可以通过综合值 $\gamma(\gamma_i=\rho_i\delta_i)$ 来确定，显然 γ 越大就越有可能是聚类中心. 由图可以看到右上方的点的分布相对下方的点来说较为离散，ρ_i 和 δ_i 的值相对较大，由此便可以选出较为合适的输入参数来确定聚类中心.

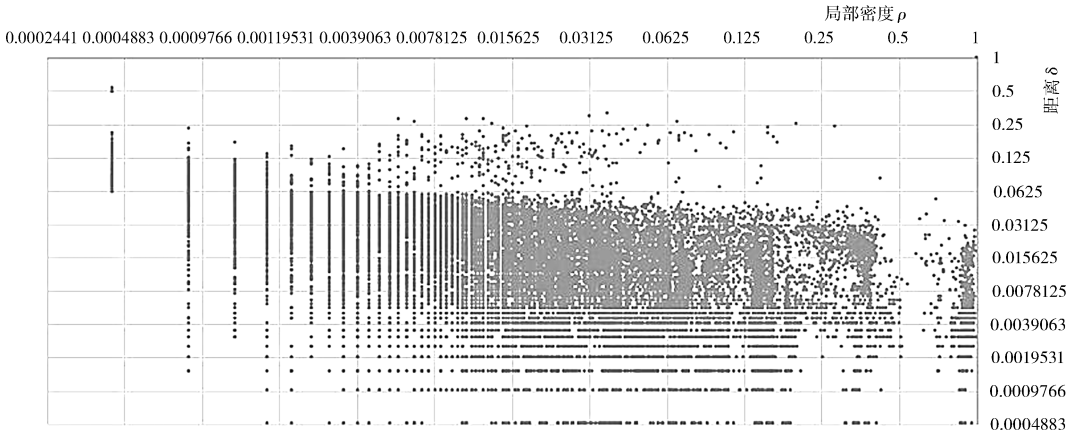

图 3 CFSFDP 算法决策图

Fig. 3 The decision graph of CFSFDP

5 结 论

本文深入研究了当前主要的恶意软件分析与聚类方法,针对存在的问题,从特征的提取、表示以及聚类算法的选择与评估方面进行改进,提出了基于模糊哈希特征表示的恶意软件聚类方法.实验结果表明,与目前的主流的几种杀毒软件相比,本文提出的方法得到的调和平均值最高且聚类个数与 VX-Heaven 标记的最为接近.从聚类效果来看,本方法产生的聚类结果,簇内紧密度高,簇内样本的动态特征较为相似,簇与簇之间有较好的区分度.在未来的工作中,我们将进一步改进自动化分析工具的能力以提取更完整的特征,同时将结合分类算法以及集成学习的方法尽可能多的减少噪声,以提升对恶意软件分类的能力.

参考文献:

[1] Akhgar B, Brewster B. Combatting cybercrime and cyberterrorism[M]. [s. l.]: Springer International Publishing, 2016.

[2] Mari Kert. CAPITAL the cybersecurity research agenda for privacy and technology challenges [DB/OL]. (2013-09-16). [2017-02-18]. www. capital-agenda. eu.

[3] Ye Y, Li T, Chen Y, *et al*. Automatic malware categorization using cluster ensemble [C]// Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. [s. l.]: ACM, 2010.

[4] Cesare S, Xiang Y, Zhou W. Malwise——An effective and efficient classification system for packed and polymorphic malware [J]. IEEE Trans Comput, 2013, 62: 1193.

[5] Kolosnjaji B, Zarras A, Webster G, *et al*. Deep learning for classification of malware system call sequences[J]. Artuf Intell, 2016, 2016: 9992.

[6] 郑荣锋, 方勇, 刘亮. 基于动态行为指纹的恶意代码同源性分析[J]. 四川大学学报:自然科学版, 2016, 53: 793.

[7] Liang G, Pang J, Dai C. A behavior-based malware variant classification technique[J]. Int J Info Edu Tec, 2016, 6: 291.

[8] Damiani E, Vimercati S D C D, Paraboschi S, *et al*. An open digest-based technique for spam detection [C]// Proceedings of the International Conference on Parallel and Distributed Computing Systems. San Francisco, California, USA: DBLP, 2004.

[9] Oliver J, Cheng C, Chen Y. TLSH——A locality sensitive hash[C] // Proceedings of the Cybercrime and Trustworthy Computing Workshop. Washington, DC, USA: IEEE Computer Society, 2014.

[10] Brun M, Chao S, Hua J, *et al*. Model-based evaluation of clustering validation measures[J]. Pattern Recogn, 2007, 40: 807.

[11] Sambasivam S, Theodosopoulos N. Advanced data clustering methods of mining web documents[J]. Issu Info Sci Info Tec, 2006, 3: 563.

[12] Rodriguez A, Laio A. Clustering by fast search and find of density peaks[J]. Sci, 2014, 344: 1492.