基于调用习惯的恶意代码自动化同源判定方法

乔延臣[1,2,3],云晓春[1,3],张永铮[3],李书豪[3]
(1. 中国科学院计算技术研究所,北京 100190; 2. 中国科学院大学,北京 100049;
3. 中国科学院信息工程研究所,北京 100093)

摘 要: 恶意代码同源判定对作者溯源、攻击事件责任判定、攻击场景还原等研究工作具有重要作用. 目前恶意代码同源判定方法往往依赖人工分析,效率低下,为此,提出一种基于调用习惯的恶意代码自动化同源判定方法. 该方法基于 7 类调用行为,使用数据挖掘算法构建作者编程习惯模型,基于频繁项离群检测算法计算同源度,利用 K 均值聚类算法选择同源判定阈值,进而实现恶意代码同源判定. 实验结果表明,该方法具有 99% 以上的准确率和可接受的召回率.

关键词: 网络安全; 恶意代码; 同源判定; 调用习惯; 自动化
中图分类号: TP393.08 **文献标识码:** A **文章编号:** 0372-2112 (2016)10-2410-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.10.019

An Automatic Malware Homology Identification Method Based on Calling Habits

QIAO Yan-chen[1,2,3], YUN Xiao-chun[1,3], ZHANG Yong-zheng[3], LI Shu-hao[3]
(1. *Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190, *China*;
2. *University of Chinese Academy of Sciences, Beijing* 100049, *China*;
3. *Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093, *China*)

Abstract: Malware homology identification is useful for malware authorship attribution, attack scenario restoration, and so on. Current malware homology identification methods still rely on manual analysis, which is inefficient and time-consuming. In order to improve the effectiveness and efficiency, an automatic malware homology identification method is proposed. Based on 7-class calling behaviors, this method constructs a model of calling habits using data mining algorithms. Then it calculates the degree of homology based on Frequent Pattern Outlier Factor. Finally, it chooses the threshold values using k-means clustering algorithm to identify homology. The experimental evaluations on real-world malwares show our method achieves high accuracy (over 99%) and acceptable recall rate.

Key words: network security; malware; homology identification; calling habits; automatic

1 引言

本文中,若两个恶意代码由同一作者或同一组织所研发则称它们同源,它们可能属于不同的家族,甚至具有很大的功能差异. 随着攻击方式向高级、持续(如 APT, Advanced Persistent Threat)等方向发展,通常一个攻击由多种恶意代码完成或不同的攻击所用的恶意代码均出自同一作者或组织,发现其中的同源关系对作者溯源、攻击场景还原、APT 攻击防范等具有重要作用. 现阶段,同源判定主要依赖人工分析,如 Sasser 与 Netsky[1]、Duqu 与 Stuxnet[2~4]、Flame 与 Gauss[5] 等的同源判定工作均由领域专家人工分析完成,虽然分析结果详细全面,可信度高,但受专家经验影响较大,因此效率较低. 针对上述问题,亟需高效地恶意代码同源判定方法.

习惯一般指在不知不觉中有规律重复的行为[6],代码易变但编程习惯不易改变,因此可以根据编程习惯进行同源判定. 在源代码作者溯源[7]中,编程习惯主要包括编程布局、编程风格、编程结构等,然而在编译过程中排版、布局、命名等均丢失,生成的二进制程序仅保

收稿日期:2015-04-05;修回日期:2015-08-24;责任编辑:李勇锋
基金项目:国家自然科学基金(No. 61303261);国家 863 高技术研究发展计划(No. 2013AA014703, No. 2012AA012803);国家 242 信息安全计划(No. 2014A094);中国科学院战略性科技先导专项(No. XDA06030200)

留了 WinAPI(Windows Application Programming Interface)调用习惯等. 同时恶意代码为达到目的往往调用大量 WinAPI,在此前提下提出了利用恶意代码蕴含的 WinAPI 调用习惯构建作者编程习惯模型以检测同源恶意代码的方法. 该方法以同源的恶意代码样本集为输入,根据样本集提取作者的7类 WinAPI 调用习惯,之后为每个样本计算同源度并选择同源度阈值,根据阈值判定待测样本是否同源.

2　相关工作

目前,恶意代码同源判定结论均依赖人工分析. CrySyS 实验室的 Bencsáth[4] 等人发现 Stuxnet 与 Duqu 存在很多相同的特殊关键词,注入机制、注入目标、导出函数、导入函数所用的特殊手法、负载与配置、通信模块等也具有相似性,基于这些证据,认为二者同源. Gostev[3] 花费2个月分析 Duqu 木马,发现 Stuxnet 与 Duqu 所用驱动文件在编译平台、时间、代码等方面具有相似性. Symantec 的 Eric Chien[2] 等人深入分析发现 Duqu 具有收集信息的功能,结合 Duqu 与 Stuxnet 在开发平台、代码等方面的相似性,总结出 Duqu 是 Stuxnet 先驱的结论,更进一步地分析出二者之间的关系. Kaspersky 实验室[8] 的专家深入分析 Stuxnet 与 Flame,发现 2009 版 Stuxnet 中的一个模块是 Flame 中的插件,得出 Flame 与 Stuxnet 的开发人员有过早期合作等结论. CrySyS 实验室的 Bencsáth[5] 在 2012 年基于详细的分析报告得出 Stuxnet 与 Duqu 同源,Flame 与 Gauss 同源. FireEye[9] 实验室 2013 年深入分析 11 个高级持续攻击(APT),在攻击所用恶意代码中发现了相同的代码段、时间戳、数字证书等,基于这些发现认为这些攻击均由一个组织操纵. 各个实验室、反病毒厂商的专家给出的分析报告详细全面,有力地证明了不同恶意代码的同源关系,但受专家经验影响较大,花费时间较长,效率较低.

源代码作者溯源主要基于作者的编程习惯. Krsul 等人[10] 将编程习惯定义为编程布局、编程风格、编程结构等三类,经实验验证分类准确率为 73%. MacDonell 等人[11] 基于 Krsul 等[10] 定义的三类编程习惯,自动化提取 C++语言蕴含的作者编程习惯用于作者溯源,分类准确率为 81.1%. Ding 等人[12] 仍然基于三类编程习惯,对 Java 语言构建不同作者的编程习惯模型用于作者溯源,分类准确率为 67.2%. 基于作者编程习惯能有效地将源代码溯源至所属作者,然而编译之后的二进制代码无排版、布局、命名等特征,无法直接利用现有方法解决恶意代码同源判定问题,但具有重要的借鉴意义.

由于人工同源判定效率低,不适应海量恶意代码样本的同源判定需求. 结合源代码作者溯源工作中基于编程习惯的思想,本文提出了基于调用习惯的恶意代码自动化同源判定方法.

3　方法描述

本文提出了基于 WinAPI 调用习惯的恶意代码自动化同源判定方法. 如图1所示,该方法包括两个阶段:同源学习阶段与同源判定阶段. 在学习阶段,基于同源样本集构建 WinAPI 调用习惯模型,依据频繁模式离群因子[13](Frequent Pattern Outlier Factor,FPOF)计算样本同源度,使用 K 均值聚类算法选择同源判定阈值. 判定阶段,提取待测样本的 WinAPI 调用行为,再结合第一步中构建的 WinAPI 调用习惯模型与同源判定阈值,判断待测样本是否同源.

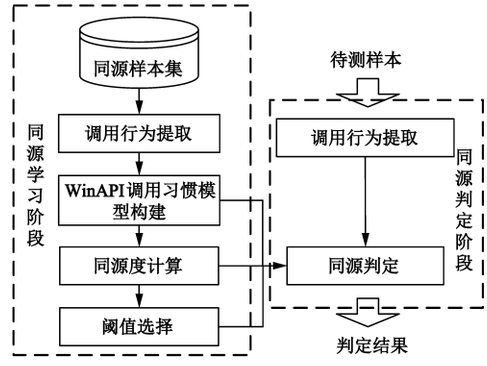

图1　基于WinAPI调用习惯的恶意代码同源判定方法步骤

3.1　调用行为定义

以习惯定义[6] 类推,在长期编程过程中不知不觉有规律重复的 WinAPI 调用行为,即 WinAPI 调用习惯. 基于编程经验与恶意代码分析经验,共总结出7类 WinAPI 调用行为. 样本(*Sample*)经逆向反汇编后获得汇编代码,依据调用指令 *call* 索引到函数,表示为 *Proc*;依据跳转指令 je、jne、jbe 等将 *Proc* 切分成多个代码段,表示为 *Loc*. 根据调用指令与跳转指令,每个 *Sample* 被划分为多个 *Proc*,每个 *Proc* 包含多个 *Loc*,结构如图2所示.

7类 WinAPI 调用行为定义如下:

B_{S2C}:*Sample* 级 2-WinAPI 组合调用行为,对2个 WinAPI,在同一样本中调用的行为;

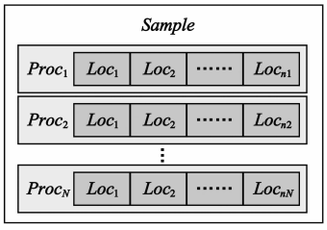

图2　*Sample*、*Proc*、*Loc*结构示意图

B_{P2C}:*Proc* 级 2-WinAPI 组合调用行为,对 2 个 WinAPI,在同一函数中调用的行为;

B_{P2S}:*Proc* 级 2-WinAPI 分离调用行为,对 2 个 WinAPI,分别在不同函数中调用的行为;

B_{L2C}:*Loc* 级 2-WinAPI 组合调用行为,对 2 个 WinAPI,在同一代码块中调用的行为;

B_{LS}:*Loc* 级 WinAPI 单独调用行为,对某个 WinAPI,在代码块中仅调用该 WinAPI 而不调用其他 WinAPI 的行为;

B_{LSeq}:*Loc* 级 WinAPI 序列调用行为,对某些 WinAPI,在代码块中依特定序列调用这些 WinAPI 的行为;

B_{L2O}:*Loc* 级 2-WinAPI 先后次序调用行为,对 2 个 WinAPI,在代码块中依特定次序调用的行为.

依据行为定义,每类行为每个样本均可提取零到多个 WinAPI 调用行为,进而生成调用行为集合,每个样本可提取 7 类调用行为集合. 样本 S 的 7 类调用行为集合可表示为:$B^S=(B^S_{S2C},B^S_{P2C},B^S_{P2S},B^S_{L2C},B^S_{LS},B^S_{LSeq},B^S_{L2O})$,每类行为集合表示为:$\{b^1,b^2,\cdots,b^m_s\}$,其中 b 表示一个行为,m_s 表示样本 S 中该类行为的总数.

3.2 同源学习阶段

该阶段,针对训练集样本提取的 7 类行为,统计其发生规律进而提取其中的频繁调用行为以构建作者的 WinAPI 调用习惯模型,基于习惯模型利用频繁模式离群因子[13]计算样本同源度,之后利用 K 均值聚类算法将训练集样本聚为两类或合并为一类,以选择同源度阈值. 该阶段结束时完成 WinAPI 调用习惯模型的构建,同时生成样本同源度计算公式、同源度阈值,以在同源判定阶段对待测样本进行同源判定.

3.2.1 习惯模型构建

首先计算调用行为的出现频率,假设训练集共 N 个样本,其中 n 个样本蕴含该行为,则该行为的出现频率为 $P_b=n/N$,当 P_b 超过一定阈值 P_{thres}(本文 P_{thres}设定为 0.9)时,认为该行为可表示作者的一个编程习惯,表示为 $h(=b)$,则 $P_h(=P_b)$ 为该习惯的稳定度,P_h 越高表示该习惯在作者编写样本中出现的频率越高. 最终形成 7 类带有稳定度的 WinAPI 调用习惯集合,为 WinAPI 调用习惯模型,表示为:$H=(H_{S2C},H_{P2C},H_{P2S},H_{L2C},H_{LS},H_{LSeq},H_{L2O})$,每类习惯表示为 $\{h^1:P^1_h,h^2:P^2_h,\cdots,h^l:P^l_h\}$,其中 h 表示其中的一个习惯,P_h 表示该习惯对应的稳定度,l 表示该类习惯的总数.

3.2.2 同源度计算

He[13]定义了频繁模式离群因子,用于检测单类数据集中的离群点. 本文借鉴频繁项离群因子,将其定义为同源度. 每个样本提取 7 类调用行为,在每类行为上依据挖掘的 WinAPI 调用习惯计算同源度,样本 S 在第 k 类行为上的同源度计算公式如下:

$$C^S_k=\frac{\sum\limits_{h\in H_k,h\in B_k}P_h}{\sum\limits_{h\in H_k}P_h} \tag{1}$$

其中 k 表示定义的 7 类行为中的 1 类,因此,$B_k\in\{B_{S2C},B_{P2C},B_{P2S},B_{L2C},B_{LS},B_{LSeq},B_{L2O}\}$,同样 $H_k\in\{H_{S2C},H_{P2C},H_{P2S},H_{L2C},H_{LS},H_{LSeq},H_{L2O}\}$,$C^S_k$ 表示样本 S 第 k 类行为下的同源度. 每个样本均计算 7 类行为的同源度($C^S_{S2C},C^S_{P2C},C^S_{P2S},C^S_{L2C},C^S_{LS},C^S_{LSeq},C^S_{L2O}$).

3.2.3 阈值选择

对训练集样本均计算 7 类同源度,假设在训练集样本中存在第 k 类行为下同源度表现不明显的样本,即该类行为下的离群点[14],利用 K 均值算法[15]将第 k 类行为下训练集样本的同源度进行 2 类聚类,形成 $Class_1$:($C^1_k,C^2_k,\cdots,C^i_k$)与 $Class_2$:($C^{i+1}_k,C^{i+2}_k,\cdots,C^N_k$)两类,其中 $C^1_k\geqslant C^2_k\geqslant\cdots\geqslant C^i_k>C^{i+1}_k\geqslant C^{i+2}_k\geqslant\cdots\geqslant C^N_k$,$N$ 表示样本总数. 选择 $Class_1$ 中的最小值做为同源判定阈值,最后选出 7 类行为下的 7 个阈值:($T_{S2C},T_{P2C},T_{P2S},T_{L2C},T_{LS},T_{LSeq},T_{L2O}$).

3.3 同源判定阶段

如图 3 所示,对每个待测样本,首先从样本中提取 7 类行为集合;其次,依据学习阶段获得的 WinAPI 调用习惯模型构造同源度计算公式,计算待测样本的 7 类同源度;最后,若其中有 t 类不小于学习阶段选取的阈值,则认为同源支持度为 $Sup=t/7$. 同源支持度越高表明同源关系越强,根据实践经验,同源支持度的阈值一般在[0.5,0.8]效果比较好,本文以 0.5 为例,当同源支持度 $Sup>0.5$ 时,认为待测样本与训练集同源. 同源判定的过程如图 3 所示.

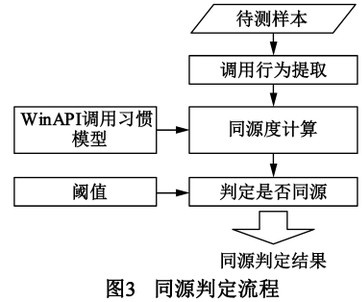

图3 同源判定流程

4 实验与分析

4.1 实验数据集

恶意代码作者一般不留作者的信息,因此明确作者的样本非常少,经详尽调研,VXHeaven[16]网站上有少量作者标注的样本. 去除调用 WinAPI 少于 10 个等无法提取有效行为的样本,经整理去重后收集 43 个具有作者标注的样本,如表 1 所示.

表 1　数据列表

作者	样本数
roy g biv	9
Vorgon	6
others	28

4.2　roy g biv 同源判定实验

作者 roy g biv 共 9 个样本,从中任取 5 个作为训练集,其余样本与其他作者的共 38 个样本做测试集,共进行了 $\binom{9}{5}=126$ 次实验.

如图 4 所示,在 126 次实验中,判定准确率为 100% 的实验有 107 次,未检出同源样本的实验有 16 次,平均准确率为 99.10%.

如图 5 所示,在 126 次实验中,判定召回率为 100% 的有 8 次,判定召回率为 0% 的有 16 次,平均召回率为 43.64%.

该同源判定实验中,平均准确率高达 99.10%,误报率低.但是召回率仅为 43.64%,漏报率比较高.

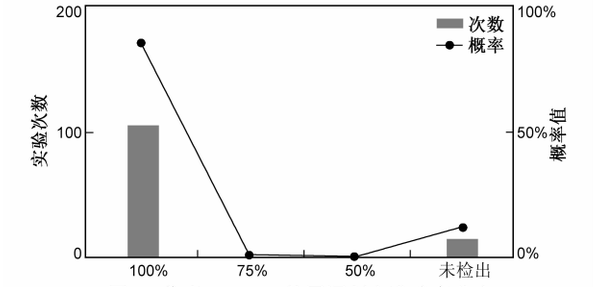

图4　作者roy g biv的同源判定准确率分布

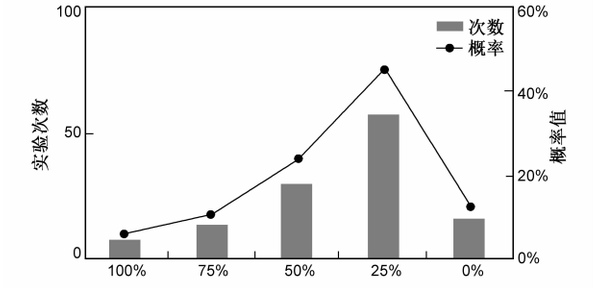

图5　作者roy g biv的同源判定召回率分布

4.3　Vorgon 同源判定实验

作者 Vorgon 共 6 个样本,从中任取 4 个作为训练集,该作者其余样本与其他作者共 39 个样本做为测试集,共进行了 $\binom{6}{4}=15$ 次实验.

图 6 中概率值 -100% 表示该次实验未检出任何同源样本.如图 6 所示,在 15 次实验中,未检出同源样本的实验有 5 次,占比 33.33%,其余 10 次实验中准确率均为 100%.Vorgon 同源判定实验的平均准确率为 100%,平均召回率为 60%.

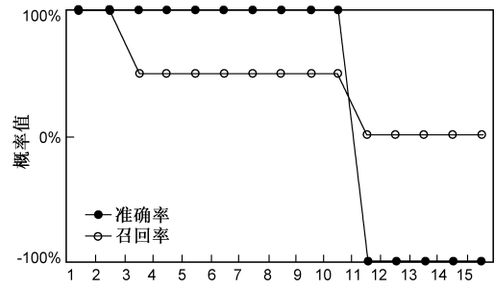

图6　作者Vorgon训练集样本数为4时的准确率与召回率

5　工作比较

Shankarapani 等人[17]提出利用序列比对方法,通过计算静态 WinAPI 调用序列间的序列相似度,进行恶意代码变种识别.本文与 Shankarapani[17]的工作进行比较,以此说明本文同源判定方法与变种识别等方法的差异,本文方法能有效判定源自同组织、作者的不同恶意代码间的同源关系.

5.1　同源判定比较

Shankarapani[17]的工作中设定的阈值为 0.5,以作者 roy g biv 的 9 个样本做对比实验.Shankarapani[17]的方法将 2 对样本 boundary 与 impute、efishnc 与 junkmail 识别为同源样本,但未发现这 4 个样本与其余 5 个样本间的同源关系.依据该判定结果,将 9 个样本划分为两类,以 boundary、impute、efishnc、junkmail 等 4 个样本为集合 A,其余 5 个样本为集合 B,基于 Shankarapani[17]的方法,集合 A 与 B 无同源关系.

以集合 A 做训练集,集合 B 中 5 个样本与其他作者样本为测试集,做同源判定实验;以集合 B 做训练集,集合 A 中 4 个样本与其他作者样本为测试集,做同源判定实验.两个实验的结果如图 7 所示,当以 A 中 4 个样本为训练集时,判定 B 中的 gemini 为同源样本;当以 B 中 5 个样本做为训练集时,判定 A 中的 impute 与 efishnc 为同源样本.

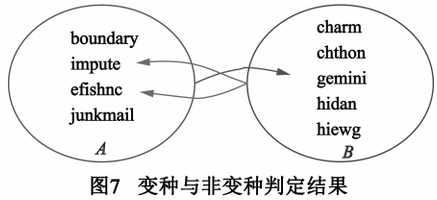

图7　变种与非变种判定结果

5.2　复杂度分析

Shankarapani[17]的方法与本文方法均需对恶意代码样本做逆向等处理,该阶段耗费时间相同.Shankarapani[17]的方法时间复杂度主要取决于最长公共子序列(LCS,Longest Common Subsequence)算法,其时间复杂度为 $O(mn)$,其中 m 与 n 分别为 2 个样本 WinAPI 序列的长度.本文方法时间复杂度主要取决于行为提

取,行为提取的时间复杂度依赖于组合行为的提取,其复杂度为 $O(k^2)$,其中 k 为样本中调用的 WinAPI 个数,经统计,样本中 WinAPI 数不大于样本的 WinAPI 调用序列长度,即 $k \leqslant m, n$. 因此,本文方法的时间复杂度较低,尤其适合在海量样本寻找同源样本.

5.3 讨论

通过对比实验不难发现,用于变种识别的方法,并不适用于同源判定工作,主要是因为同源判定工作同时包含了不同变种、不同家族间恶意代码的同源判定,而不同家族恶意代码的功能、行为等差别较大,导致 WinAPI 调用序列间的最长公共子序列较小. 但本文方法基于不易变的编程习惯,能有效判定功能、行为差别较大的恶意代码间的同源关系.

本文方法尚有局限性:一是需要同源的多个样本做训练集以构建作者的习惯模型进而实现同源判定,Shankarapani[17] 的方法仅需 1 个样本即可进行变种判定;二是本文方法需要依据多类行为,Shankarapani 的方法仅需依据样本的完整 WinAPI 调用序列即可完成变种判定. 此外,两个方法均具有很高的准确率.

6 总结

针对现有恶意代码人工同源判定方法效率低下的问题,本文提出了基于 WinAPI 调用习惯的恶意代码自动化同源判定方法. 实验结果表明,同一作者编写的不同恶意代码蕴含相同的 WinAPI 调用习惯,利用 WinAPI 调用习惯进行同源判定是可行的. 基于频繁项离群点检测算法能很好地识别训练集中的离群点并统计出判定阈值,以 99% 以上的准确率与可接受的召回率识别出同源样本,为海量恶意代码同源判定提供技术支撑.

参考文献

[1] 董志强,肖新光,张栗伟. 编码心理学分析病毒同源性[J]. 信息安全与通信保密,2005,2005(8):55 - 59.

[2] Chien E, Omurchu L, Falliere N. W32 Duqu: the precursor to the next stuxnet[A]. Proceedings of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)[C]. Berkeley, CA: USENIX Association, 2012. 5 - 5.

[3] Gostev A, Soumenkov I. Stuxnet/Duqu: The evolution of drivers[OL]. http://www. securelist. com/en/analysis/204792208/Stuxnet Duqu The Evolution of Drivers, 2011.

[4] Bencs TH B, P K G, Butty N L, et al. Duqu: A Stuxnet-like malware found in the wild[R]. CrySyS Lab Technical Report, 2011.

[5] Bencs TH B, et al. The cousins of stuxnet: Duqu, flame, and gauss[J]. Future Internet, 2012, 4(4): 971 - 1003.

[6] Butler G, Hope R A. Manage Your Mind: The Mental Fitness Guide[M]. Oxford University Press, 2007.

[7] Burrows S, Uitdenbogerd A L, Turrin A. Comparing techniques for authorship attribution of source code[J]. Software: Practice and Experience, 2014, 44(1): 1 - 32.

[8] Lab K. Resource 207: Kaspersky Lab Research Proves that Stuxnet and Flame Developers are Connected[OL]. http://www. kaspersky. com/about/news/virus/2012/Resource_207_Kaspersky_Lab_Research_Proves_that_Stuxnet_and_Flame_Developers_are_Connected, 2012.

[9] Moran N, Bennett J. Supply chain analysis: From quartermaster to sunshop[J]. FireEye Labs, 2013, 11: 1 - 39.

[10] Krsul I, Spafford E H. Authorship analysis: Identifying the author of a program[J]. Computers & Security, 1997, 16(3): 233 - 257.

[11] Macdonell S G, Gray A R, Maclennan G, et al. Software forensics for discriminating between program authors using case-based reasoning, feedforward neural networks and multiple discriminant analysis[A]. Proceedings of the 6th International Conference on Neural Information Processing (ICONIP'99)[C]. Perth, WA: IEEE, 1999. 66 - 71.

[12] Ding H, Samadzaden M H. Extraction of Java program fingerprints for software authorship identification[J]. Journal of Systems and Software, 2004, 72(1): 49 - 57.

[13] He Z, Xu X, Huang Z J, et al. Fp-outlier: frequent pattern based outlier detection[J]. Computer Science and Information Systems/ComSIS, 2005, 2(1): 103 - 118.

[14] He Z, Xu X, Huang J Z, et al. A frequent pattern discovery method for outlier detection[A]. Advances in Web-Age Information Management[M]. Springer, 2004. 726 - 732.

[15] Krishna K, Murty M N. Genetic K-means algorithm[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 1999, 29(3): 433 - 439.

[16] Heaven V. Computer virus collection[OL]. http://vxheaven. org/vl php, 2014.

[17] Shankarapani M K, Ramamoorthy S, et al. Malware detection using assembly and API call sequences[J]. Journal in Computer Virology, 2011, 7(2): 107 - 119.

作者简介

乔延臣 男,1988 年 9 月出生,山东聊城人. 2010 年毕业于山东大学数学学院,获学士学位. 现为在读博士生. 主要从事网络信息安全、恶意代码等方面的研究工作.
E-mail: qiaoyanchen@ iie. ac. cn

云晓春 男,1971 年 2 月出生,黑龙江哈尔滨人. 1999 年获哈尔滨工业大学获工学博士学位. 现为中国科学院计算技术研究所研究员、博士生导师,主要从事信息安全、计算机网络等方面的研究工作.
E-mail: yunxiaochun@ cert. org. cn