一种基于混合学习的恶意代码检测方法

梁光辉[1,3],摆 亮[2],庞建民[1,3],单 征[1,3],岳 峰[1,3],张 磊[4]
(1. 解放军信息工程大学,河南郑州 450002; 2. 国家计算机网络应急技术处理协调中心,北京 100029;
3. 数学工程与先进计算国家重点实验室,河南郑州 450002; 4. 78090 部队,四川成都 610000)

摘 要: 近年来,自动化沙箱被广泛部署并应用于恶意代码分析与检测,然而随着恶意代码数量的激增和抗分析能力的增强,如何有效应对海量恶意代码分析任务,提高沙箱系统分析效率,是增强网络安全防御能力的一个重要研究方向. 本文利用不同学习方式以及恶意代码动、静态特征的特点,提出了一种基于混合学习模型的恶意代码检测方法,分别提取恶意代码的静态模糊哈希特征和动态行为特征,然后将无监督聚类学习与有监督的分类学习相结合用于恶意代码检测. 实验表明,在不影响检测精度的情况下,只利用了原有系统 0.02% 分析时间提高了整个系统 25.6% 的检测速度.

关键词: 恶意代码; 模糊哈希; 混合学习
中图分类号: TP305 **文献标识码:** A **文章编号:** 0372-2112 (2021)02-0286-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20180711

A Malware Detection Method Based on Hybrid Learning

LIANG Guang-hui[1,3], BAI Liang[2], PANG Jian-min[1,3], SHAN Zheng[1,3], YUE Feng[1,3], ZHANG Lei[4]
(1. *PLA Information Engineering University, Zhengzhou, Henan* 450002, *China*;
2. *National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing* 100029, *China*;
3. *State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, Henan* 450002, *China*;
4. *Unit* 78090, *Chengdu, Sichuan* 610000, *China*)

Abstract: In recent years, automated sandboxes have been widely deployed for malware analysis and detection. However, with the rapid increase column of malware and the enhancement of anti-analysis capabilities, how to effectively handle these massive malware analysis tasks and improve the efficiency of sandbox system is an important research topic. Based on the characteristics of different learning methods and malware dynamic and static features, this paper proposes a malware detection method based on a hybrid learning model. We extract static fuzzy-hash features and dynamic behavior features of malware, then unsupervised clustering learning is combined with supervised classification learning. Experiments show that using only 0.02% of the analysis time improves the detection speed of the entire system by 25.6% without affecting the detection accuracy.

Key words: malware; fuzzy hash; hybrid learning

1 引言

据赛门铁克 2017 年安全威胁年报统计,相比于 2016 年,移动端恶意代码增长了 54%,物联网恶意代码增长了 600%,勒索软件增长了 46%[1]. 快速增长的恶意代码给信息系统安全带来了巨大的破坏和损失,同时也给分析与防御工作带来了很大的挑战. 研究人员根据已有分析技术提出了各种检测方法,如静态特征匹配、启发式检测、动态行为监控等,这些方法都在一定程度上缓解了海量恶意代码分析所面临的压力. 随着机器学习和人工智能的发展,恶意代码的检测与分析进入了一个新的阶段,大量基于机器学习模型的恶意代码检测方法不断出现[2~4]. 为了更加准确的捕获恶意代码的行为,各种蜜罐和自动化沙箱开始不断被开发出来并推广,从 2010 年至今,先后出现了多个典型的沙箱系统,如 Ether、CWsandbox、Anubis、Temu、Cuckoo[5~8] 等,这些自动化沙箱通过捕获目标软件在运行期间的

收稿日期:2018-08-09;修回日期:2020-01-16;责任编辑:王天慧
基金项目:国家自然科学基金(No. 61472447, No. 61802435, No. 61802433)

行为轨迹,能够有效的发现未知恶意行为,提高判定的准确度.

在基于自动化沙箱的恶意代码分析实践上,一方面为了监控程序的整个运行过程,在每一个分析周期内,沙箱需要尽可能运行更长时间以便采集到尽可能丰富的信息来建立行为轮廓,另一方面虚拟化技术需要消耗大量的系统资源来模拟各种软硬件,过长时间的运行将是一笔较大的资源开销. 尤其是某些处于关键网络节点的沙箱系统,每天需要分析成百上千的可疑代码,这些可疑代码往往都被无差别地输入到沙箱系统中进行分析. 沙箱的分析时间一般设定在 2min 到 5min,有些甚至更长. 这种无差别的动态分析无疑是对沙箱分析资源的一种浪费,当任务队列较长或对分析的实时性要求较高时,这种动态分析方式也在一定程度上延缓了安全威胁被发现的时机. 所以,很多研究人员提出了加速沙箱分析的方法,例如基于实时动态轨迹的威胁性判断,当判定为恶意时,就立即终止当前沙箱分析任务. 这种方法在一定程度上提高了动态沙箱的效率,但是这种方法一般是针对某种特定类型的样本,例如和命令服务器连接的下载器或者间谍软件等,适用性不够强. 因此,本文提出了基于混合分析的恶意代码检测方法,技术路线如图 1 所示.

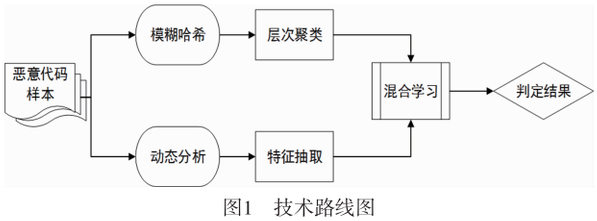


图1　技术路线图

恶意代码样本进入系统后,分别经过模糊哈希和动态分析两种,通过模糊哈希计算恶意代码的相似度,并基于层次聚类获得恶意代码的簇类关系;通过动态分析获取恶意代码的动态系统调用序列,并抽取对应的动态特征,两种处理结果通过混合学习模型之后,实现对批量恶意代码的分析与检测,在不影响检测精度的情况下,只利用了原系统 0.02% 分析时间提高了整个系统 25.6% 的检测速度.

2　相关工作

Matthias 等人提出了一种行为预测模型,利用已有的动态行为聚类模型结果,预测可疑软件的行为模式[9]. Bayer 等人[10]认为大量新出现的恶意软件很大程度上都是已发现的恶意软件的变形或者重打包. 所以可以通过对这些变形或加壳的恶意软件的识别提高动态分析的效率. Phani 等人[11]认为当前的动态执行框架存在一个问题,就是所有样本都会被执行,作者认为这是一种资源浪费,所以提出了概率多假设测试框架 MAXS,通过威胁情报和流量等信息来判断一个将要执行的样本是否是被重打包的样本.

基于此,本文提出了基于混合学习的恶意代码检测方法. 具体来说,本文的贡献如下:

(1)提出了基于混合学习的恶意代码检测框架,融合了无监督的聚类方法和有监督的分类方法,进一步提高了动态分析的处理能力;

(2)提出了基于程序代码段的模糊哈希聚类方法,并基于聚类结果设计了相应的抽样算法;

(3)在实际的恶意代码测试集上测试并验证了混合学习模型的检测效率.

3　基于模糊哈希的聚类分析

模糊哈希最初被大型邮件系统用来识别垃圾邮件,因为邮件正文的大小相对比较稳定,不会存在较大的波动,如 ssdeep 算法[12]. 后来为了处理较大体积或者比较对象之间的文件大小差距比较明显时,例如图像、文档类文件时,Roussev 和 Breitinger 等人提出了 sdhash 和 mvhash[13,14].

Tridgell[15]定义了模糊哈希算法的两个基本属性:非传播特性和对齐鲁棒性. 传统哈希算法对文件内容的变动十分敏感,文件中一个细微的改变会传播到计算的全过程,从而完全改变哈希结果. 模糊哈希中的非传播特性是指对文件中任何位置的微小改变,仍然可以保留文件其他部分的特征还在最终哈希结果中. 对齐鲁棒性是指改变文件的对齐结果,例如插入或删除一个字符,模糊哈希能够重新生成对齐结果,不会影响最终的模糊哈希计算结果. 非传播特性和对齐鲁棒性保证了模糊哈希算法能够保证两个文件在具有一定差异的情况下生成较为相似的哈希值.

3.1　主流模糊哈希算法

(1)Mvhash

Mvhash 称为基于多数投票的模糊哈希(Fuzzy Hashing Using Majority Vote),分为 mvhash-L 和 mvhash-B. mvhash-B 投票算法,将输入转换为等长度的 0X00 或者 0XFF 字节序列,然后用 RLE 编码压缩输入,之后再利用改进的 bloom 过滤器生成等长的特征(2018bit),作为最终的哈希特征. 两个输入之间的相似度通过汉明距离来计算.

(2)Sdhash

Sdhash 利用了亚马逊书籍推荐中的特征选择方法,选取了文件中的 SIP(Statistically-Improbable-Phase)特征,并利用特征的熵过滤掉了识别度较低的特征,这在一定呈上降低了误报率,最后通过布隆过滤器生成了需要对比的文件的相似性哈希.

(3) ssdeep

ssdeep 也称为基于上下文触发的分段哈希(Context Triggered Piecewise Hash),是加密哈希(Cryptographic Hash),滚动哈希(Rolling Hash)和分段哈希(Piecewise Hash)的组合. 通常情况下,用于分片的是一个弱哈希算法,同时还有一个强哈希算法用来计算每个分片的哈希,之后通过一个压缩映射算法,将计算后的分片哈希值映射为一个更短的值,最后通过比较算法对生成的模糊哈希进行计算,得到两个模糊哈希的相似程度.

3.2 基于模糊哈希相似度的层次聚类

Google 公司的 VirusTotal 网站在 2012 年以后开始采用模糊哈希对恶意代码进行标记[16]. 模糊哈希虽然具有对齐鲁棒性,但是当比较的文件大小之间存在较大差距时,会对比较的结果产生一定的影响,即两个文件大小差距较大,如果小文件是大文件的一部分,最终的相似度结果会受文件大小的影响. 本文研究的主要是二进制恶意代码,二进制可执行程序往往包含多个段,如代码段、数据段、资源段等. 具有相同功能的程序,代码段往往相同,但资源段或数据段却可能存在较大差异,而且程序的代码段保存了二进制程序的主要语义信息,存储着程序的指令序列,所以为了更好的利用模糊哈希的计算效果,本文对程序的代码段进行模糊哈希计算.

在无监督的学习中,本文使用了层次聚类算法. 层次聚类通过层次的架构方式,反复将数据进行聚合或者分裂,形成一个层次序列. 层次聚类分为层次聚合算法和层次分裂算法,本文使用层次聚合算法,在聚合过程中,两个类之间距离的相似性度量方法采用上节中代码段的模糊哈希相似度. 层次聚类中的联接规则采用平均联接规则. 虽然层次聚类的复杂度为 $o(n^2)$,但是相比于动态沙箱分析的时间而言,层次聚类对整个系统的效率产生的影响非常小.

4 基于混合学习的检测方法

4.1 动态行为特征

本文的动态行为特征生成与提取主要借助于 Cuckoo 沙箱和 *N*-gram 算法. Cuckoo 沙箱 2012 年由谷歌“Summer Code”计划推出,经过多年的维护和不断升级,已经成为当前主流的开源沙箱. 为了更好模拟真实主机环境,在 Guest 操作系统上安装了 office、Adobe、影音播放软件等常见的应用软件.

特征向量生成上,本文采取了 3-Gram 特征生成方法. 该特征生成方法近年来在恶意代码检测中被广泛应用,并且取得了较好的检测效果. 本文将多个进程合并的行为轨迹作为 API 的行为序列流后,然后利用滑动窗口在行为序列流上滑动,窗口的大小为 3.

每个样本的动态行为特征向量表示如下:

$$\boldsymbol{B}=(gs_1,gs_2,\cdots,gs_n)$$

其中gs_i表示第 i 个 3-Gram 片段,gs_1的取值为 0 或 1,0 表示不存在这个行为序列片段,1 表示存在这个行为序列片段. 例如对于某一段系统调用序列{NtOpenKey, NtQueryValueKey, NtClose, NtUserSetCursor, NtAllocateVirtualMemory, NtGdiHfontCreate},当滑动窗口为 3,滑动间距为 1 时,其对应的特征向量为{(NtOpenKey, NtQueryValueKey, NtClose), (NtQueryValueKey, NtClose, NtUserSetCursor), (NtClose, NtUserSetCursor, NtAllocateVirtualMemory), (NtUserSetCursor, NtAllocateVirtualMemory, NtGdiHfontCreate)},本文实际实验中,N 大小取 3,滑动间距为 1.

4.2 基于混合学习的恶意性判定算法

本文使用的有监督学习的模型为随机森林,该模型相对于单个分类器相比具有更好的分类效果和泛化能力. 在 4.1 节生成的 3-gram 特征将作为随机森林模型的特征输入,本文主要目的是为了验证混合学习的检测效率,所以在随机森林模型的参数选择上都采取了缺省默认值,并没有进行调优. 其中随机森林的分类器数量为 64,随机森林训练完成之后的判定模型定义为如下函数:

$$Flag=\text{Random\_Forest}(T_1) \tag{1}$$

T_1为待检测的未知恶意代码,返回值为 *Flag*,当 *Flag* 为 1 时判定为恶意,为 0 时判定为良性.

假设要检测的未知样本集合 $\boldsymbol{S}=\{s_1,s_2,\cdots s_n\}$,共包含 n 个样本. 首先利用 3.2 节中的模糊哈希对样本集合 S 进行聚类分析,假设在阈值 γ 的时候($0<\gamma<1$),$\boldsymbol{S}$ 集合的聚类结果为 $C=(c_1,c_2,\cdots,c_k)$,C 中包含 k 个簇($1\leqslant k\leqslant n$). 对于聚类之后的 k 个簇,本文采取反馈式聚类抽样算法,来选择最终输入动态沙箱的样本. 传统的聚类抽样是将每一个簇看作一个独立的单位,一个簇要么被抽取,要么不被抽取. 本文根据聚类后簇的样本分布情况确定抽样比例,按照每个簇的样本比例对该簇进行随机抽样,抽样过程为不放回抽样,抽样函数定义如下:

$$T=\text{Random\_select}(c_i,m) \tag{2}$$

其中,c_i指具体的某个簇,m 是从该簇中需要随机抽取的样本数,T 为随机抽样后返回的样本集合.

综上,基于聚类结果的恶意性预测算法如下算法 1 所示:

算法 1 基于聚类结果的恶意性预测算法

Input: Each cluster c_i in $C=(c_1,c_2,\cdots,c_k)$

Output: Malware detection result of samples in each cluster

1: $m=\lfloor \log_2 len(c_i) \rfloor$

```
2:  T = Random_select(c_i, m)
3:  Flag = Random_Forest(T_1)
4:  Second_sampling = False
5:  for T_j in T
6:    if Flag ≠ Random_Forest(T_j)
7:      Second_sampling = True
8:      break
9:    endif
10: endfor
11: if Second_sampling = = False
12:   Return Flag
13: endif
14: U = Random_select(c_i - T, 2m)
15: Flag = Random_Forest(U_1)
16: for U_j in U
17:   if Flag ≠ Random_Forest(U_j)
18:     Return Error
19:   endif
20: endfor
21: Return Flag
```

Second_sampling 为是否进行二次抽样的标志位,当 *Second_sampling* 为 True 时,则二次抽样的样本集合为初始的 2 倍. 当算法返回的结果为正常的 *Flag* 时,表明随机抽样集合的判定结果一致,则该簇所有的样本都被标记为该 *Flag* 对应的类别,当返回的结果为 Error 时,表明抽样集合的判定结果不一致,则该簇的所有样本都将进行动态特征抽取并判定恶意性. 在随机抽样中,当第一次抽样的判定结果不一致时,会进行第二次抽样,第二次抽样的样本数为前一次的两倍,当第二次抽样的判定结果仍然不一致时,则返回 Error,如果结果一致,则返回对应的 *Flag*.

5　实验结果与分析

文中的恶意样本从 Malware Benchmark 恶意代码库中选取了从 2012 年到 2018 年共计 51695 个二进制样本. 良性样本来自于两部分,第一部分为纯净版 Windows 7 系统下的可执行程序,第二部分为大型程序安装后的可执行程序组件,良性样本总数为 4713. 实验平台的操作系统为 Ubuntu 16.04,32G 内存,1T 硬盘,动态沙箱 Cuckoo 版本为 2.0,Guest 系统为 Windows7 32 位操作系统.

5.1　模糊哈希聚类速度分析

本文选取 50,000 个恶意代码作为模糊哈希聚类的测试样本,图 2 为样本的聚类结果图. 横坐标为样本的数量,纵坐标为层次聚类所需时间,单位为秒. 通过图 2 的结果可以发现,三种聚类算法中,mvhash-b 的计算速度最快,最慢的为 sdhash,ssdeep 的计算速度居中,随着样本数量的增多,mvhash-b 的计算速度几乎是 sdhash 的两倍. 同时,和动态沙箱分析相比,模糊哈希分析的速度要远远快于动态沙箱分析.

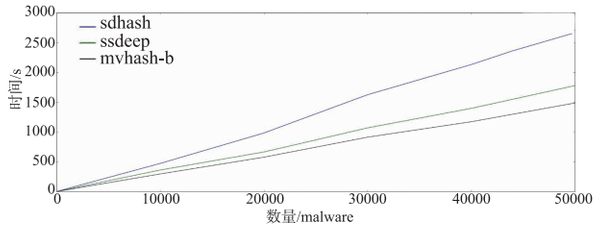


图2　模糊哈希算法聚类速度

5.2　模糊哈希的聚类准确度分析

对于包含有 n 个恶意代码的输入集合 $S=\{s_1,s_2,\cdots s_n\}$ 的参考聚类结果为 $RC=(rc_1,rc_2,\cdots,rc_t)$,模糊哈希的聚类结果为 $O=(o_1,o_2,\cdots,o_k)$,则聚类的准确率(Precision)定义如下:

$$\text{Precison}=\frac{1}{n}\sum_{i=1}^{t}\max(|rc_i\cap o_1|,rc_i\cap o_2|,\cdots,rc_i\cap o_k|)$$

同时,召回率(Recall)的定义如下:

$$\text{Recall}=\frac{1}{n}\sum_{j=1}^{k}\max(|rc_1\cap o_j|,rc_2\cap o_j|,\cdots,rc_t\cap o_j|)$$

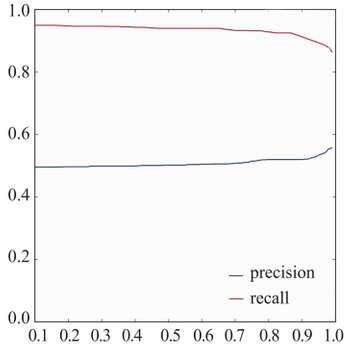
本文中使用的参考聚类 结果 RC 来自于 AVCLASS 生成的聚类结果,该方法比较几十款杀毒软件对恶意代码的命名,通过投票算法计算出恶意代码的家族信息[17].

图 3 的 3(*a*)、3(*b*)、3(*c*),分别为 sdhash、ssdeep、mvhash 三种模糊哈希的聚类结果图,横坐标为聚类的阈值,纵坐标为准确率和召回率的值. 可以看到在聚类的准确率方面,相对来说 ssdeep 要好于 sdhash 和 mvhash,在聚类的召回率方面,sdhash 要好于 ssdeep 和 mvhash. 三种模糊哈希整体上都表现为较高的召回率,较低的准确率. 这说明模糊哈希聚类的特点是当样本之间存在较大相似度时都能将其准确聚类. 这种聚类模型精确度虽然不是最高,但是具有较低的误报率,这种特性也是后面进行恶意性预测的基础.

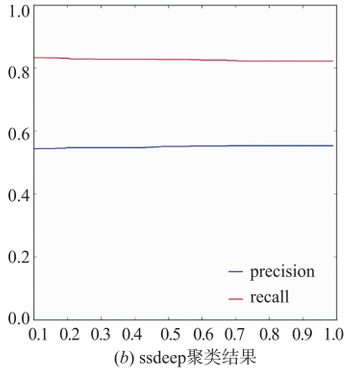
5.3　恶意性检测

本文选取了 10000 个恶意样本,4700 个正常样本,恶意样本与正常样本的比率大致为 2∶1,采用了十倍交叉验证的方法来分配训练集和测试集. 在同样的样本集合下,分别进行了两次实验,第一次是基于动态行为特征的恶意性检测,第二次是基于混合学习的恶意性检测,混合学习中,本文选取了在 5.2 节中的 ssdeep 来计算样本的模糊哈希.

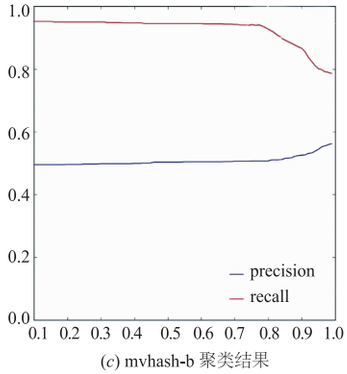
图 4 是两种检测模型的 ROC 曲线对比图,可以发现基于动态行为特征的检测模型 ROC 值为 0.932,基于混合学习模型的检测模型 ROC 值为 0.925,两种检测模型的 ROC 值差距仅为 0.07. 同时,两种检测模型的其他检测指标如表 1 所示.



(*a*) sdhash聚类结果



(*b*) ssdeep聚类结果



(*c*) mvhash-b 聚类结果

图3

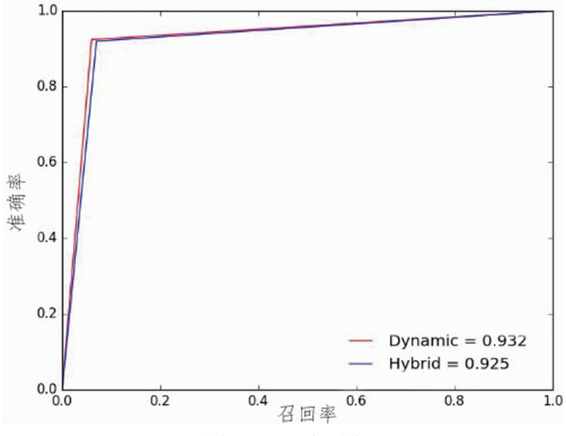


图4 ROC对比图

表 1 检测模型对比结果

指标 \ 检测模型	Dynamic	Hybrid
准确率	0. 9685	0. 9633
召回率	0. 924	0. 92
F1 值	0. 945	0. 941

可以发现,基于动态行为特征的检测模型和混合学习模型在准确率、召回率和 F1 值等方面的差别较小,也就是说混合学习检测模型和动态行为检测模型具有相同的恶意代码检测能力. 同时,本文统计了混合学习模型在十次交叉验证的测试阶段,每次需要送入沙箱提取特征的样本数量. 如表 2 所示.

表 2 混合学习中沙箱分析样本数量统计

实验序号	1	2	3	4	5
分析样本数	1026	1107	1214	1006	1149
实验序号	6	7	8	9	10
分析样本数	1121	1047	1056	1109	1027

在动态行为检测模型中,每一个测试样本都需要送入动态沙箱提取行为特征,也就是 1470 个测试样本都需要进行动态分析,但是,在混合学习检测中,需要送入沙箱提取动态特征的样本数量平均仅为 1086 个. 也就是说,混合分析系统通过模糊哈希聚类(需要 11s)节省了 384 个样本的动态分析时间,如果采用原有动态分析的话,这 384 个样本需要 $384 \times 120 = 46080$s,相比来说,模糊哈希只占了动态分析 384 个样本所需时间的 0. 02%.

从整体上分析,混合学习在保持检测率不变的情况下,相比于动态行为检测模型,在只占用 0. 02% 的分析时间的情况,提高了整个系统 25. 6% 的分析速度. 本文所提出的模型已经在国家计算机网络应急技术处理协调中心的检测系统上应用并部署,在运行过程中,较大的提升了当前已有系统的分析效率.

6 总结

本文从当前恶意代码数量快速增长下动态沙箱资源消耗过大的问题出发,提出了基于混合学习的恶意性检测模型,借助于恶意代码静态分析和动态分析的特点,提出了基于静态模糊哈希特征的无监督聚类学习和基于动态行为特征的有监督分类学习,在不影响检测率的前提下,有效提高了基于沙箱的动态检测效率,在恶意代码测试集上验证了本方法的有效性,并在实际的应用环境中检验了本方法的实用性和可扩展性.

未来工作中,可以将有监督学习扩展到多分类学习模型,在检测恶意性的基础上能够进行家族判定,这样的

判定结果可以进一步反馈给无监督学习的聚类模型,指导聚类结果的优化,进一步的提升检测的准确度和效率.

参考文献

[1] Symantec Corporation. Executive Summary-2018 Internet Security Threat Report [EB/OL]. https://www. symantec. com/content/dam/symantec/docs/reports/istr-23-executive-summary-en. pdf,2018-03-01/ 2018-04-12.

[2] Bayer U,Comparetti P M,Hlauschek C,et al. Scalable,behavior-based malware clustering[A]. ISOC. NDSS[C]. San Diego,USA:ISOC ,2009. 8 – 11.

[3] Willems C,Holz T,Freiling F. Toward automated dynamic malware analysis using cwsandbox[A]. IEEE Security & Privacy[C]. Oakland,USA:IEEE. 2007. 32 – 39.

[4] Malware Benchmark. [EB/OL]. http://www. malwarebenchmark. org,2016-10-25/2018-05-07.

[5] Sebastián M,Rivera R,Kotzias P,et al. Avclass:A tool for massive malware labeling[A]. International Symposium on Research in Attacks,Intrusions,and Defenses[C]. Cham,Swit:Springer,2016. 230 – 253.

[6] Mao WX,Cai ZM,Tong L. Malware detection method based on active learning[J]. Journal of Software,2017,28(2):384 – 397.

[7] Dinaburg A,Royal P,Sharif M,et al. Ether:malware analysis via hardware virtualization extensions[A]. Proceedings of the 15th ACM conference on computer and communications security[C]. Alexandria,USA:ACM,2008. 51 – 62.

[8] Willems C,Holz T,Freiling F. Toward automated dynamic malware analysis using cwsandbox[A]. IEEE Security & Privacy[C] Oakland,USA:IEEE,2007. 26 – 32.

[9] Neugschwandtner M,Comparetti P M,Jacob G,et al. Forecast:skimming off the malware cream[A]. Proceedings of the 27th Annual Computer Security Applications Conference[C]. Orlando,USA:ACM,2011. 11 – 20.

[10] Bayer U,Kirda E,Kruegel C. Improving the efficiency of dynamic malware analysis[A]. Proceedings of the 2010 ACM Symposium on Applied Computing[C]. Switzerland:ACM,2010. 1871 – 1878.

[11] Vadrevu P,Perdisci R. MAXS:Scaling malware execution with sequential multi-hypothesis testing[A]. Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security[C]. Xi'an,China:ACM,2016. 771 – 782.

[12] Kornblum J. Identifying almost identical files using context triggered piecewise hashing[J]. Digital Investigation,2006,3:91 – 97.

[13] Roussev V. Data fingerprinting with similarity digests[A]. IFIP International Conference on Digital Forensics[C]. Berlin,GER:Springer,2010. 207 – 226.

[14] Breitinger F,Astebøl K P,Baier H,et al. Mvhash-b-a new approach for similarity preserving hashing[A]. IT security incident management and it forensics (IMF)[C]. Nuremberg,GER:IEEE,2013. 33 – 44.

[15] ATridgell Spamsum Readme,[EB/OL] http://www. samba. org/ftp/unpacked/junkcode/spamsum/,2011-02-05/2017-10-14.

[16] Sarantinos N,Benzaïd C,Arabiat O,et al. Forensic malware analysis:The value of fuzzy hashing algorithms in identifying similarities[A]. Trustcom/BigDataSE/I-SPA[C]. Tianjin,China:IEEE,2016. 1782 – 1787.

[17] Sebastián M,Rivera R,Kotzias P,et al. Avclass:A tool for massive malware labeling[A]. International Symposium on Research in Attacks,Intrusions,and Defenses[C]. Evry,France:Springer,2016. 230 – 253.

作者简介

梁光辉　男,1987 年 6 月出生,陕西兴平人,现为信息工程大学博士研究生. 主要研究方向为机器学习与恶意代码分析.

E-mail :lghray@ gmail. com

摆　亮　男,1983 年5 月出生,北京人,清华大学,博士,现为国家互联网应急中心高级工程师. 研究方向为物联网与关键基础设施安全.

E-mail :bailiang@ cert. org. cn

庞建民(通信作者)　男,1964 年 4 月出生,河北沧州人,英国杜伦大学博士,现为信息工程大学教授、博士生导师. 研究方向为网络安全、先进计算.

E-mail :jianmin_pang@ 126. com