情报工程 ISSN: 2095-915X
第1卷 第6期 90-101 2015年12月
TECHNOLOGY INTELLIGENCE ENGINEERING
ISSN: 2095-915X Vol.1 No.6 90-101 Dec 2015

doi:10.3772/j.issn.2095-915x.2015.06.013

一种基于机器学习的分布式恶意代码检测方法

董立勉[1]，左晓军[1]，曲武[2]，王莉军[3]

（1. 国网河北省电力公司电力科学研究院 河北 050021；2. 北京启明星辰信息安全技术有限公司核心研究院 北京 100193；
3. 中国科学技术信息研究所 北京 100038）

摘要：随着恶意代码规模的快速增长，传统的恶意代码检测方法已经逐渐失效。主流的启发式技术由于其动态执行的特点，在许多应用中难以推广。因此，通过恶意代码的静态特征，将可疑恶意代码PE文件分类到相应的恶意代码家族是相当重要的。在本文，基于对恶意代码PE文件的分析结果，提出元数据的概念，并于此实现了恶意代码快速检测原型，PE-Classifier。在spark分布式环境中，通过使用随机森林分类算法，基于恶意代码元数据，能够对恶意代码进行快速和精准地分类和检测。实验结果表明，通过对大量的恶意代码PE样本元数据分析，本文提出的原型系统PE-Classifier能够根据元数据相似性判断样本的语义相似性，从而辅助检测，使得反病毒软件更为有效。

关键词：恶意代码检测，随机森林算法，元数据，Spark

中图分类号：TP391

A distributed Approach to Fast Malware Classification Based on Machine Learning

DONG LiMian[1], ZUO XiaoJun[1], QU Wu[2], WANG LiJun[3]

(1.Hebei Electric Power Research Institute, Hebei,050021, China; 2. Core Research Institute, Beijing Venustech Cybervision Co. Ltd., Beijing 100193, China; 3. Institute of Scientific and Technical Information of China, Beijing 100038)

Abstract: With the rapid increase of malware, conventional malware detection approaches increasingly fail,

作者简介：董立勉：（1968.10-），女，河北省石家庄市，本科，高工，国网河北省电力公司电力科学研究院，主要研究发向信息系统建设及信息安全；左晓军 （1973.12-），男，河北省石家庄市，硕士，高工，国网河北省电力公司电力科学研究院，主要研究方向：信息安全、网络管理、软件开发；曲武：（1981.08-），男，黑龙江省大庆市人，博士后，研究方向为网络安全、大数据、数据挖掘、自然语言理解 CCF 会员；邮箱：quwu_ustb@163.com；电话：13810641696；通信地址：北京市海淀区中关村软件园21号楼启明星辰大厦；王莉军：（1978.10-），女，博士，研究方向为大数据、情报分析。

modern heuristic technologies often perform dynamically, which is not possible in many applications due to related effort and the scale of files. Therefore, it is important for malware analysis that it is classifying unknown malware files into malware families in order to characterize the static malware characteristic accuracy. In this paper, we introduce a distributed approach to perform fast malware classification based on gene metadata of malware PE executable. We use a machine learning technique called random forest algorithm to classify malware fast and accurately. The results of large scale classifications show that our prototype system successfully determined some semantic similarity between malware according to the gene metadata, and make the antivirus more effective.

Key words: Malware classification, random forest algorithm, meta data, spark

1 引言

1983 年，Cohen 第一次形式定义了计算机病毒 [1]，并逐渐演进为恶意代码检测。恶意程序，通常被定义为恶意代码，被分类为病毒 (Virus)、特洛伊木马 (Trojans)、蠕虫（Worms）、逻辑炸弹 (Logicbomb) 和其他种类。当前，恶意代码仍然是计算机系统和网络最主要的威胁之一。根据 2014 年赛门铁克 (Symantec) 报告，244 封邮件中至少有 1 封邮件带有恶意代码附件，而其中仅有 13.9％能够被识别和阻断。而这些恶意代码中，大部分是基于以前恶意代码变种而来，变种行为通常使用工具进行修改而成，自动化程度很高，几乎可以批量生产。这些变种恶意代码可以轻易的绕过反病毒软件，实现免杀目的。仅当变种恶意代码签名被添加到反病毒软件时，这些恶意代码才能够被查杀。显然，基于签名的商业反病毒软件对于恶意代码变种或 0-day 攻击几乎是无效的。

当前，恶意代码研究已经开始聚焦于更加通用和可扩展的特征，这些特征能够替代或补充传统的恶意代码签名识别恶意代码变种或 0-day 攻击。其中，这些研究主要包括两类方法，静态分析和动态分析。静态分析，无需运行恶意代码，仅通过分析 PE 文件的代码统计信息和程序结构进行分析。动态分析，需要在模拟环境或真实环境中运行恶意代码，通过分析其恶意行为进行检测。

在本文，通过对恶意代码 PE 样本（本文提到的恶意代码特指恶意代码 PE 样本）大规模统计分析的基础上，提出元数据的概念，并于此实现了恶意代码快速检测原型 PE-Classifer。在 Spark 分布式环境中，PE-Classifer 提取恶意代码和良性代码的元数据，并使用分布式随机森林算法训练分类器。最后，基于分类器对 PE 文件进行快速和精准地分类。本文的贡献描述如下：

1）基于恶意代码 PE 文件的元数据，使用随机森林算法训练分类模型，最后自动执行恶意代码快速检测，准确分类恶意代码和良性代码；

2）为了处理大规模恶意代码样本，包括元数据的提取和模型训练。本文基于 Spark 分布式环境，实现了分布式元数据的提取和模型训练，解决了特征提取模块和模型训练模块的横向扩展问题；

3）PE-Classifer 原型系统作为传统反病毒软

件的补充，提升反病毒软件的检测能力；

4）PE-Classifer 原型系统有助于研究人员了解恶意代码 PE 文件内部的语义信息，可进一步用于恶意代码家族分类。

本文组织如下：第 2 节介绍该领域的相关工作，第 3 部分展示了 PE-Classifer 原型系统的检测原理及方法论。第 4 部分进行实验评估和结果展示。最后，进行总结和未来展望。

2 相关工作

2.1 恶意代码检测

在恶意代码检测研究领域，按照检测方法所在的位置可分为基于主机的检测和基于网络的检测 [2]。基于主机的检测技术又可分为特征签名技术、校验和技术、启发式方法和机器学习方法。基于网络检测技术可分为 DPI 技术和蜜罐的检测技术。而根据恶意代码的运行状态，检测技术又可以分为静态检测技术和动态检测技术。静态检测技术对恶意代码样本进行分析判断其是否为恶意，对于特征签名提取技术和校验和判别技术都可归为此类。在动态检测技术中，恶意代码在虚拟机或仿真器中虚拟执行获得恶意代码的主机行为特征，并进行相关语义分析来判别样本是否具有恶意行为。

传统基于签名的恶意代码检测方法难以适应恶意代码变种检测的需求。当前，许多研究人员开始聚焦于使用数据挖掘算法进行未知恶意代码检测，以此寻找恶意代码与特征之间的关联关系。为了表示恶意代码样本，研究人员广泛使用 n-gram[3][4] 或 API 调用 [4][5][6][7][8] 作为主要特征进行分析，进而训练分类器进行检测。n-gram 算法最早用于文本分类和信息检测中，核心思想是计数连续 n 个词出现的频率。Tony 等人 [9] 首先提出使用 n-gram 特征的差异性进行恶意代码检测。该方法通过自动统计算法从恶意代码 PE 文件中自动提取 ByteCode 特征，然后训练分类器进行检测。Henchiri 等人 [10] 提出一类新的 n-gram 特征提取方法。首先，该方法首先对恶意代码 PE 文件进行扫描，统计 n-gram PE 文件中出现的频率。接下来。根据已标注的恶意代码家族建立一个符合频率支持的 n-gram 特征列表，通过恶意代码家族 n-gram 特征的差异性，即过滤掉一个恶意代码家族出现较为频繁但在其他家族出现较少的特征，从而获得更具有普适性的特征，并在此基础上区分恶意代码和良性代码。Moskovitch 等人 [11] 提出利用 OpCode 代替 ByteCode 进行统计获得恶意代码的特征，实验结果表明该特征比 ByteCode 更为有效。孔德光 [12] 提出使用多维特征，包括 Opcode 序列、调用流图特征、系统调用序列图，对恶意代码家族特征进行统计和分析，并结合语义特征来表述恶意代码的“行为”特征，从而对分类结果加权投票后给出恶意代码的分类信息。

2.2 分布式处理框架

为了实现 PE-Classifer 原型系统，需要统筹考虑存储、计算和建模框架。对于存储框架，本文选择 HDFS 来管理 PE 文件元数据，HDFS 是一个主 / 从架构，从终端用户的视角看，它与传统的文件系统一样，完全可以使用目录对文件进行增删改查。但是，由于其分布式存储的特征，HDFS 集群被设计成拥有一个 NameNode 节点和若干个 DataNode 节点。NameNode 负责管理文件系统的描述元数据，DataNode 负责存储实际的数据文件。用户通过与 NameNode 和 DataNodes 的交互访问文件系统。该过程分为两个阶段，用户访问 NameNode 节点以获取文件的元数据，而真正的文件 CRUD（Create、Read、Update 和 Delete）操作时直接与 DataNode 节点进行交互的。

Spark 是 UC Berkeley AMP 实验室实现的 Map-Reduce 并行计算框架，拥有基于 Hadoop 的 Map-Reduce 实现所具有的优点，但不同于 Hadoop 的 Map-Reduce 实现，Spark Job 的中间输出和结果可以保存在内存，从而中间过程无需读写 HDFS，很大程度上降低了磁盘 IO 的开销。因此，Spark 框架能更好的适用于数据挖掘与机器学习等需要迭代的 Map-Reduce 算法。在基于 Spark 计算框架建模过程中，本文主要使用 Spark 上一个机器学习算法库 MLLib，借助 Spark 的内存计算能力，使得机器学习的模型计算时间大大缩减。

为了解析和获取恶意代码的元数据特征，本文使用 Pandas，Scikit-learn 等 Python 数据分析包对恶意代码的 PE 文件进行分析，获得的恶意代码元数据。

3 PE-Classifier 检测原理

本节，恶意代码检测的原型系统 PE-Classifier 的检测原理将会被详细描述。通过引入数据挖掘技术，PE-Classifier 能够精确地检测新的恶意代码及其变种。图 1 显示了恶意代码检测系统 PE-Classifier 的架构，核心模块包含分布式特征提取模块、特征选择和变换模块以及基于 Spark 的随机森林分类模块：

1) 分布式特征提取模块：基于 Spark 并行计算框架，该模块对于 PE 文件进行批量分析获取 56 个特征，形成 56 维的特征向量；

2) 特征选择和变换模块：基于 PCA 算法对 56 维特征向量进行处理，降低维度；

3) 基于 Spark 的随机森林分类算法。

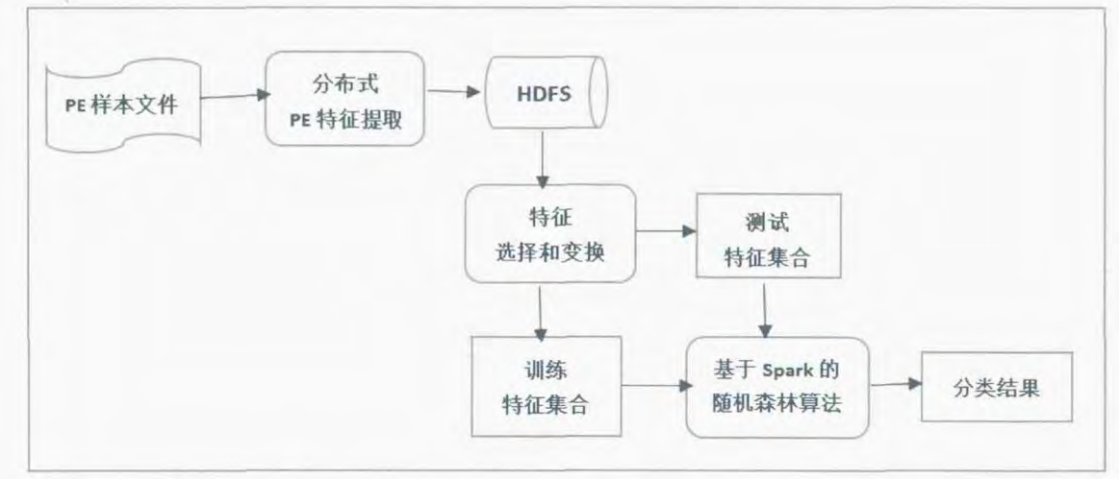


图 1 PE - Classifier 框架

3.1 检测原理

本文提出的方法，其核心检测原理是恶意代码二进制 PE 文件包含的特征标识可充分用于恶意代码自动识别和分类，其原理假设描述如下：

1) 为了绕过 AV 的签名检测，恶意代码通过变种来剔除或规避签名，但恶意代码 PE 文件包含的特征在一定程度上是稳定的；

2) 由于恶意代码的工作原理导致其 PE 文件特征和结构与源于商业开发环境的良性代码差异较大，这种差异仅体现在功能上，并不影响其执行能力。而且，这种差异在一定程度上能够用来区分恶意代码与良性代码；

3) 通常，恶意代码会采用加壳、加密和其他混淆方法绕过 AV 的查杀；

4) 为了快速开发以及快速变种，低水平的

编程技术通常会引发 PE 文件结构的异常；

5) 为了实现恶意代码常用的功能，一些 API 函数的关键组合通常会被包含到 PE 文件中；

6) 恶意代码作者留在代码里的注释等信息，这类信息通常为文本字符串。

基于以上 6 点原理假设，本文的研究和验证工作主要包含以下 4 个步骤，分别为潜在的有效特征设计和提取、特征约简、特征贡献评估和系统评估。潜在的有效特征设计和提取，基于文献 [13][14][15][16][17][18][19] 的研究成果，及其本文的研究结论，56 个特征元数据被从 PE 文件中提取出来用于恶意代码识别，并使用 F56 来表示该集合。为了使本文的特征能够覆盖恶意代码不同区段的特征，不同区段的特征都会被进行提取和分析，描述如下：

1）来自 PE 文件头 (DOS 头和 PE 头）的文件外部结构信息，以及来文件结构内部的区段信息都将被提取；

2）全局特征，例如文件的熵值或包含特定的字符串，函数特征，例如程序流重定向扫描或相关引入函数的统计信息。

总之，56 个潜在的特征从以上描述中进行提取。为了移除不适合的特征，构建恶意代码 PE 文件元数据，第二阶段特征约简描述如下。特征约简，为了评估以上 56 个特征在恶意代码分类方面的有效性，本文将为特征评估进行建模，具体建模过程将在 3.2 节进行描述。特征实用性评估，基于一个训练数据集，本文进行特征区贡献度评估，用于评价 56 个特征在分类恶意代码和良性代码样本方面的能力，即分类贡献度。该过程结果是一个统计模型，该模型描述了各个特征对于分类的贡献度，并最终生成恶意代码 PE 文件的元数据。系统评估，为了验证恶意代码 PE 文件的元数据在恶意代码分类方面的有效性，基于 Spark 分布式环境，本文实现了一个原型系统 PE-Classifier，对测试 PE 文件集合进行分类，结果为恶意代码和良性代码。基于 2 个测试数据集，具体描述见 4.1 节。

3.2 特征贡献度评估

在形式化模型过程中，为描述恶意代码和良性代码的分布，定义 $P(s)$ 为恶意代码 s 分布比例，$1-P(s)$ 为良性代码 N 分布比例。对于一个特征 A_j，令 $a_{i,j}$ 表示属性值 j 在特征 i 同时出现的情况，其发生的概率 $P(a_{i,j})$ 定义：

$P(a_{i,j}) = Ps(a_{i,j}) * P(s) + PN(a_{i,j}) * (1 - P(s))$。对于给定的恶意代码样本，条件概率为

$P(s \mid a_{i,j}) = \frac{Ps(a_{i,j})P(s)}{P(a_{i,j})}$。基于以上定义，对于特征的评估方法的 $D(a_{i,j})$ 主要由恶意代码的总概率和条件概率差异的绝对值进行定义，

$D(a_{i,j}) = \mid P(s \mid a_{i,j}) - P(s) \mid = \mid \frac{Ps(a_{i,j})p(s)}{P(a_{i,j})} - P(s) \mid$。其中，为条件下 s 的概率。最后，恶意代码特征的有效性评估 $Q(A_i)$，$0 \leqslant Q(A_i) \leqslant 100$，被定义为如下：

$Q(A_j) = 200 * [P(a_{i,1}) * D(a_{i,1}) + P(a_{i,2}) * D(a_{i,2}) + \ldots + P(a_{i,n}) * D(a_{i,n})]$。其中，泛化因子为 $100/P(s)$，$P(s)$ 默认值为 0.5，即对恶意代码和良性代码无偏倚，所以泛化因子为 200。

3.3 特征选择和特征描述

为了展示特征有效性评估，使用 PE 文件的 check_sum 作为评估指标。对于用于训练的恶意代码样本和良性代码样本，接近 75% 的恶意代码的 check_sum 为 0，而良性样本的 check_sum 分布式比较均匀，具体的分布如图 2 所示。对于 generated_check_sum 特征（如图 3 所示），恶意代码和良性样本在 generated_check_sum 的分布上难以区分，而对于 number_of_imports 特征（如

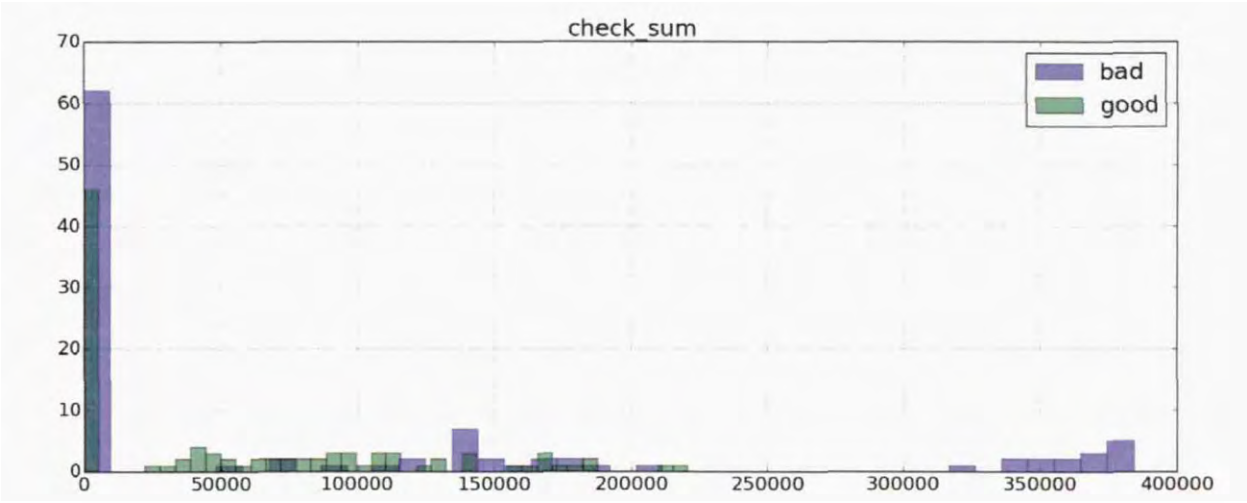


图 2 对于 check_sum 特征，恶意代码和良性代码分布

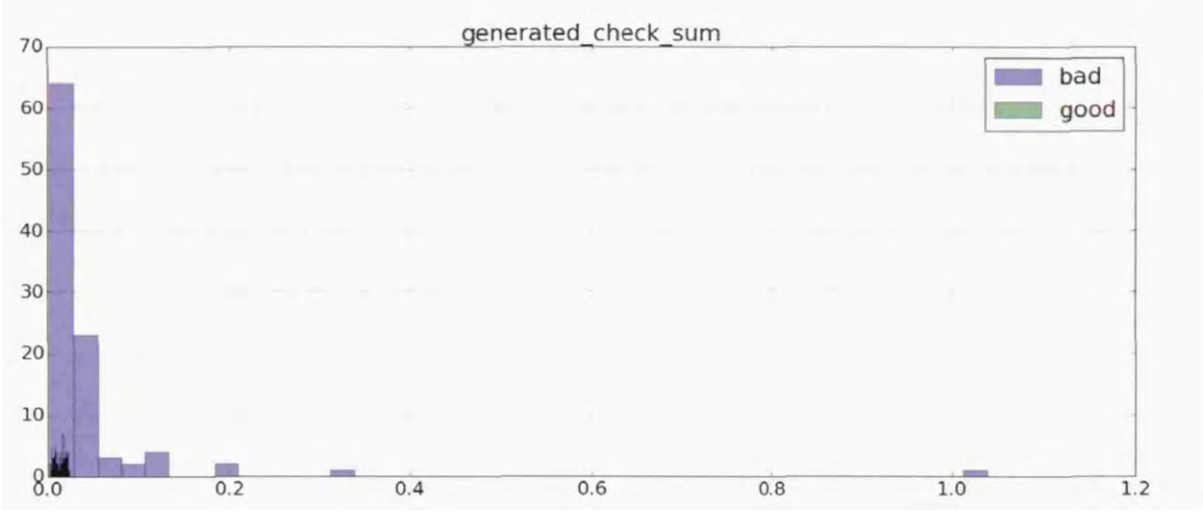


图 3 对于 generated_check_sum 特征，恶意代码和良性代码分布

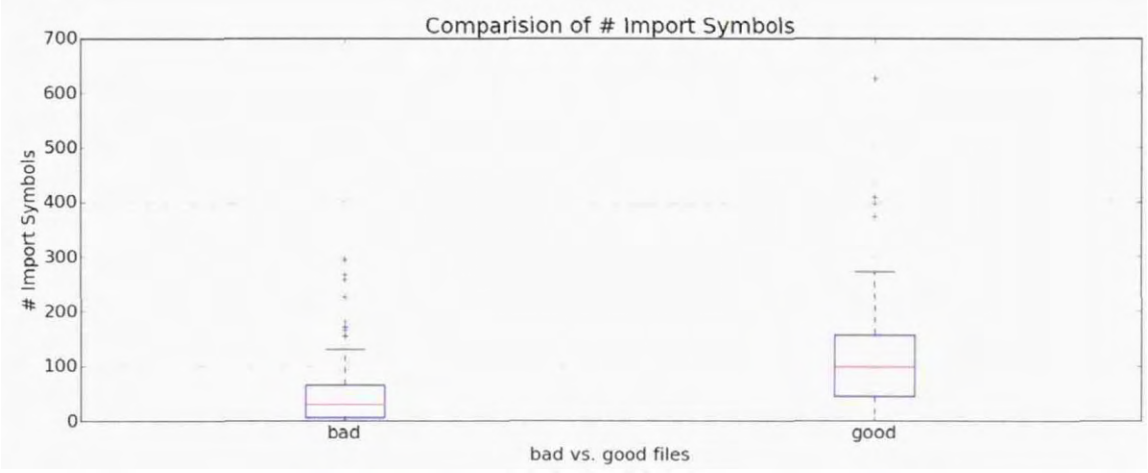


图 4 Import 特征，恶意代码和良性代码分布

图 4 所示），恶意代码和良性样本在统计方面的差异如图 3 所示，恶意代码的 number_of_imports 特征最小值为 9、最大值为 145、均值为 31，良性代码的 number_of_imports 特征最小值为 48、最大值为 148、均值为 112。显然，check_sum 特征在恶意代码分类和检测方面，其有效性优于 number_of_imports 特征和 generated_check_sum 特征，而 number_of_imports 特征优于 generated_check_sum 特征。

在分析来自训练集 PE 文件（包括恶意文件和良性文件）的 56 个特征之后，对于每个特征的有效性进行评估，使得特征在恶意文件和良性文件区分度方面更为有效。例如，对于 number_of_imports 特征（如图 4 所示）和 number_of_sections 特征（如图 5 所示），其关系如图 6 所示，恶意代码的 number_of_imports 特征多集中 80 以内，number_of_sections 特征多集中在 5 以内，而良性代码与其差异较大。在 PE 文件中，参照 number_of_sections 特征，定义为 。对于 3 个陪集，有效性评估计算描述如下：number_of_sections 特征为 0~3 的并现概率为 0.38，4~7 的并现概率为 0.55，8 以上的并现概率为 0.29。在以上特征展现情况下，PE 文件为恶意代码的概率分别为 0.78、0.27 和 0.79。因此，$Q(A_i)$ 的计算结果为 50。

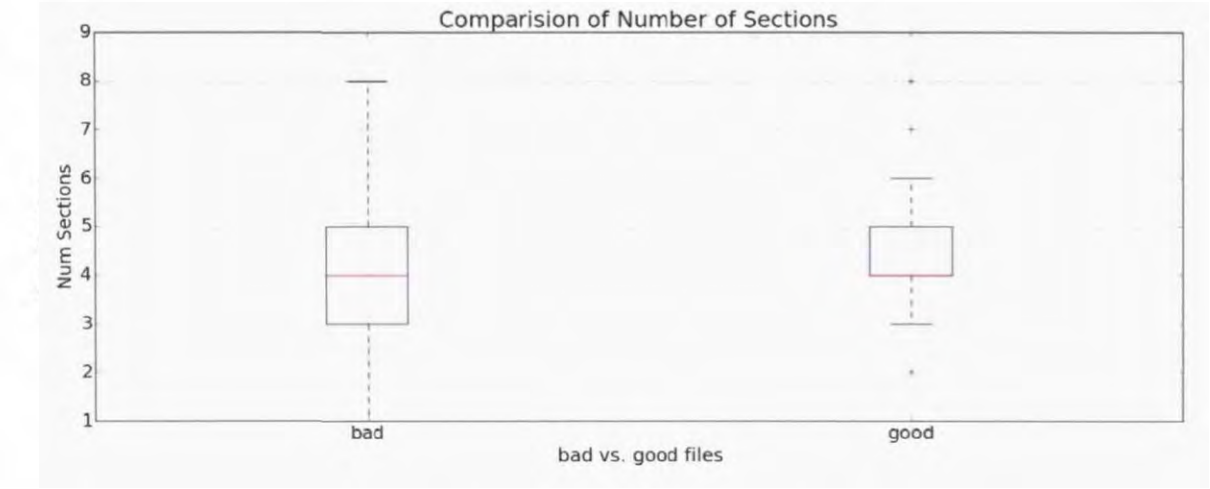


图 5 Number of Sections 特征，恶意代码和良性代码分布

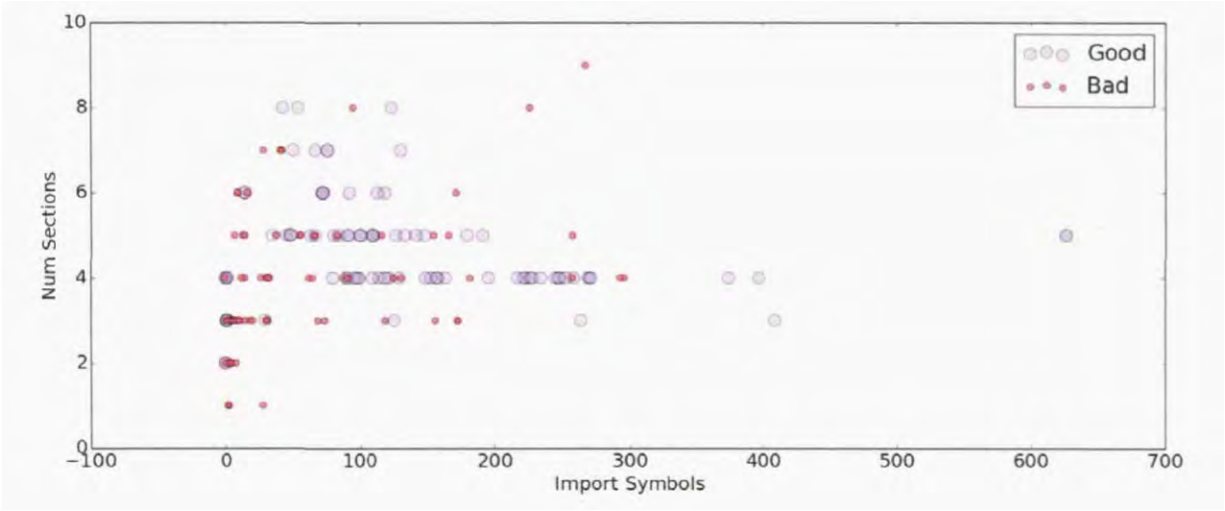


图 6 Import vs Number of sections，恶意代码和良性代码分布

3.4 基于假设检验的分类机制

根据测试样本的类别分配（恶意 / 良性）和元数据特征，本文建立了一个基于假设检验的分类机制。为此，PE 文件的元数据的特征分布已经被计算来判别样本为恶意代码的概率 $P(s \mid a_{i,j})$，具体描述见上节。即，使用训练数据集，对于 PE 文件元数据的每个特征，概率 $P(s \mid a_{i,j})$ 将被计算，并依据分类结果与事实情况的差异来判别该特征对于恶意代码检测的贡献度。为分类 PE 样本，56 个样本特征被提取构成元数据，根据测试样本（训练集和测试集），对元数据特征的不同组合进行研究，根据结果来判别分类精度。为了获得稳定的、序列独立的分类结果，训练好的分类器将使用假设检验进行评估，即引入空值假设（样本都是良性）和备择假设（样本都是恶意代码）。Ronny [20] 等人对假设检验方法进行了理论和实验评估，其中对于点的权重评估系统被认为是最优的方法。对于指定的特征描述，恶意代码的概率 $P(s \mid a_{i,j})$ 高于 S_c，S_c=60%，一个点会对每个特征额外授予 1%，每个特征能够获得最大 40 个点的评分。例如，在当前样本文分析过程中，特征 number_of_sections 被评估如下，样本包含 2 个 PE 段，特征值为 1（0~3 段）。正如在上文描述，包含该特征的恶意代码概率 $P(s \mid a_{i,j})$ 为 0.78，恶意代码概率超出阈值 S_c 为 0.18，意味着 18 个点被添加到样本总评分。通过该方式，56 个 PE 样本特征被分配点数并记入总分，高点数表明恶意代码的高概率。通过对大规模的样本分析结果，在 56 个恶意代码元数据字段中，最高评分能够达到 81 个点。

3.5 基于分类机制的统计模式识别机制

为了具体比较和评估恶意代码元数据字段对分类结果的贡献度，本文引入了基于 Spark 的机器学习库 MLlib，并使用其中的分布式分类算法 Random Forest 进行测试和评估。MLLib 是运行在 Spark 上一个机器学习算法库，借助 Spark 的内存计算，可以使机器学习的模型计算时间大大缩短。通过使用 10 交叉验证方式，评估结果表明，56 个元数据字段中，其中 10 个字段对于恶意代码分类贡献明显。

4 实验评估和结果展示

4.1 测试环境

为了评估原型系统 PE-Classifier 分类的普适性，在恶意代码元数据的设计过程中，56 个待选特征被分析和评估，其过程包含 3 个阶段：

1) 理论分析阶段：理论评估和实验分析表明，56 个待选特征中 12 个被评价为“无效 "，即对恶意代码和良性代码没有区分效果或区分效果不明显；

2) 特征有效性评估：正如在上文描述的过程，特征的有效性评估算法对余下的 44 个特征进行计算，该过程在特征提取阶段进行。对于 $Q(A_i) \geqslant 16$ 的特征被认为是有助于分类的特征。通过该阶段的筛选，接近 17 个特征被删掉，剩下 27 个特征进入第 3 阶段；

3) 误报率评估：对于 27 个特征，通过特征选择方式，即选择 n 个特征和 n-1 个特征来比较分类结果，对于导致误报率高的特征将被剔除，该过程将会保留 10 个最优的特征构成精简恶意代码元数据。

经过以上过程，使用 2 个数据集对原型系统 PE-Classifier 进行评估，测试数据集 T1 包含 11200 个恶意代码和 17900 个良性代码文件，测试数据集 T2 包含 8200 个来自网络的最新恶意代码，并且 TI 和 T2 的交集为空。

4.2 检验和评估方法

为了评估本文提出的系统，使用的基础评估指标描述如下：

1) True Positive（真正，TP）：被模型预测为正的正样本，或者可以称作判断为真的正确率；

2) True Negative（真负，TN）：被模型预测为负的负样本，或者可以称作判断为假的正确率；

3) False Positive（假正，FP）：被模型预测为正的负样本，或者可以称作误报率；

4) False Negative（假负，FN）被模型预测为负的正样本，或者可以称作漏报率；

为了评估系统的检测精度，使用 TPR（True Positive Rate）、FPR（False Positive Rate）、Precision、F-Measure、OA 四个指标进行测评：

1) TPR，表示恶意代码能被检出的概率，定义为 TPR=TP/(TP+FN)。其中，TP 表示正确检测到的恶意代码规模，FN 表示被判定为良性程序的恶意代码规模；

2) FPR, 表示良性代码被检测为恶意代码的概率，定义为 FPR=FP/(FP+TN)。其中，FP 表示被判定为恶意代码的良性代码规模 ,TN 表示为被正确认定为良性代码的规模。

3) Precision，表示被判定为恶意代码的数据中真正恶意代码所占的比例，定义为 precision=TP/(TP+FP)；

4) OA，表示检测系统的总体精度，定义为

$$OA=\frac{TP+TN}{TP+TN+FP+FN}*100\%$$

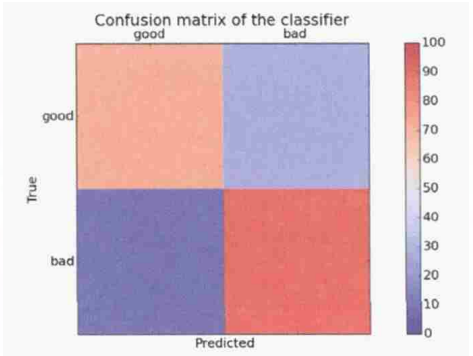
5) F-Measure，用来综合评估 Precision 和 Recall（召回率 , 与 TPR 计算相同），是 p 和 R 的加权调和平均值。定义为

$$F-Measure=[a^2+1]*P*R/[a^2*(P+R)]$$

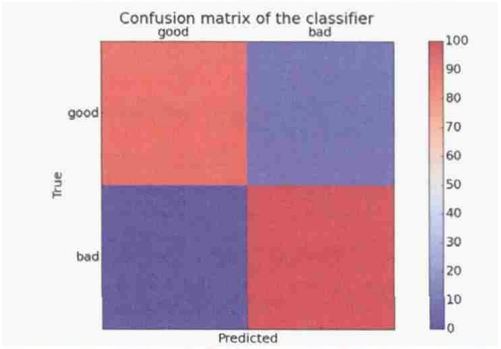
其中 P 为准确率，R 为召回率。当参数 a 取 1 时，表示最为常见的 F1-Measure，即 F1=2PR/(P+R)。

4.3 结果分析

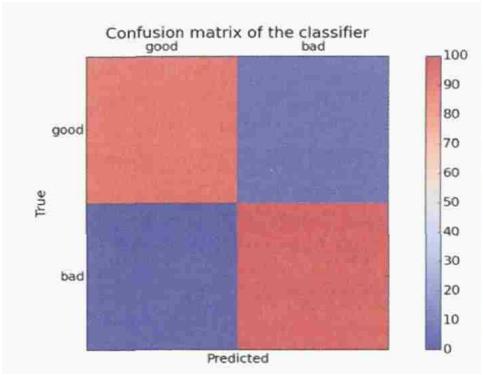
在 PE-Classifier 原型测试过程中，测试数据集 T1 被切分为训练和测试数据集。其中，采用训练数据集包含 8780 个恶意代码，12089 个良性代码，总计 3.43 GB。测试数据集包含 2420 个恶意代码和 5811 个良性代码，总计 2.07GB。在测试数据集 T1 情况下，对于 PE-Classifier 原型的 10 交叉验证的结果如图 7 所示，特征集合包括 F2、F56、F10 和 F10-Opt。其中，F2 表示特征 check_sum 和 number_of_imports。F56 表示采用所有 56 个特征对分类器进行测试。F10，包括特征 compile_date、pe_majorlink、sec_rawptr_reloc、debug_size、sec_vasize_reloc、datadir_IMAGE_DIRECTORY_ENTRY_RESOURCE_size、sec_rawptr_rdata、datadir_IMAGE_DIRECTORY_ENTRY_IAT_size、pe_char、sec_rawsize_text。F10-Opt，表示在采用 F10 特征集合的基础上，对分类器的参数进行优化和调节，获得更好的检测能力和泛化能力。由表 1 和表 2 的结果可见，在 F10-Opt 情况下，PE-Classifier 获得最好检测能力。对于测试数据集 T2，测试结果如表 3 所示。实验结果表明，在 F10-Opt 情况下，PE-Classifier 获得最好的检测精度和泛化能力。



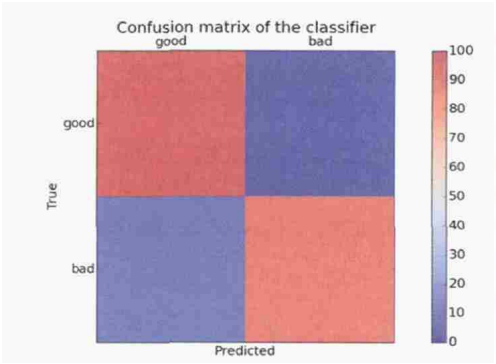
子图 1. 特征集合为 F2 混淆矩阵



子图 2. 特征集合为 F56 混淆矩阵



子图 3. 特征集合为 F10 混淆矩阵



子图 4. 特征集合为 F10 - Opt 混淆矩阵

图 7　不同的特征集合对应的混淆矩阵

表 1　基于 Spark 的随机森林分类器的实验结果

F-set	F-num	TP	TN	FP	FN
F2	2	71.11%	88.57%	28.89%	11.43%
F56	56	88.89%	97.14%	11.11%	2.86%
F10	10	91.11%	97.14%	8.89%	2.86%
F10-Opt	10	94.28%	100%	5.71%	0

表 2　PE - Classifier 的评测指标

F-set	F-num	TPR(%)	FPR(%)	Pre(%)	OA(%)	F-1
F2	2	86.15	24.6	71.11	79.84	1.106
F56	56	96.88	10.26	88.89	93.02	1.608
F10	10	96.96	8.38	91.11	94.13	1.677
F10-Opt	10	1	5.4	94.29	97.14	1.789

表 3　PE - Classifier 的泛化能力评估，T2 数据集

F-set	F-num	TPR（%）
F2	2	63.12%
F56	56	89.17%
F10	10	90.04%
F10-Opt	10	91.48%

总结和未来展望

本文基于大数据分析技术，采用新颖的恶意特征提取方法获得了恶意代码元数据，综合考虑元数据中的 56 个特征对于区分恶意代码和良性代码的有效性，获得了精简的元数据集合，并依此提出一种基于元数据的恶意代码检测方法，实现了一个恶意代码检测系统原型 PE-Classifier。实验

结果表明，通过大数据技术的引入和足够的样本规模，使得本文提出的方法能够更为准确分析和选取有效特征，PE-Classifier在恶意代码检测方面更为有效。本文的工作包含以下3个贡献：

1）恶意代码元数据的提取和有效性验证方法；

2）基于大数据分析框架Spark，使得PE文件的元数据的提取和学习更为高效，使得PE-Classifier能够分布式处理大规模的PE文件；

4）提出并实现了恶意代码检测原型PE-Classifier，使得恶意代码检测能够快速和分布式进行，检测准确率能够作为传统AV检测的有力补充。

在未来的工作中，研究提取沙箱中恶意代码的行为特征，例如主机行为、网络行为等，将恶意代码静态特征和行为特征整合起来训练分类器，使得PE-Classifier在恶意代码检测精度方面更为准确。同时，采用更大的训练和测试集合，进一步验证系统的有效性。

参考文献

[1]Cohen F. Computer viruses: theory and experiments[J]. Computers & security, 1987, 6(1): 22-35.

[2]Idika N, Mathur A P. A survey of malware detection techniques[J]. Purdue University, 2007, 48.

[3]Santos I, Penya Y K, Devesa J, et al. N-grams-based File Signatures for Malware Detection[J]. ICEIS (2), 2009, 9: 317-320.

[4]Santos I, Brezo F, Ugarte-Pedrero X, et al. Opcode sequences as representation of executables for data-mining-based unknown malware detection[J]. Information Sciences, 2013, 231: 64-82.

[5]Tabish S M, Shafiq M Z, Farooq M. Malware detection using statistical analysis of byte-level file content[C]//Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics. ACM, 2009: 23-31.

[6]Anderson B, Quist D, Neil J, et al. Graph-based malware detection using dynamic analysis[J]. Journal in Computer Virology, 2011, 7(4): 247-258.

[7]Elhadi A A E, Maarof M A, Osman A H. Malware detection based on hybrid signature behaviour application programming interface call graph[J]. American Journal of Applied Sciences, 2012, 9(3): 283.

[8]Elhadi A A E, Maarof M A, Barry B I A. Improving the detection of malware behaviour using simplified data dependent api call graph[J]. International Journal of Security and Its Applications, 2013, 7(5): 29-42.

[9]Abou-Assaleh T, Cercone N, Kešelj V, et al. N-gram-based detection of new malicious code[C]//Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International. Kyoto, Japan IEEE, 2004: 41-42.

[10]Henchiri O, Japkowicz N. A feature selection and evaluation scheme for computer virus detection[C]//Data Mining, 2006. ICDM'06. Sixth International Conference on. Hong Kong, Chian IEEE, 2006: 891-895.

[11]Moskovitch R, Feher C, Tzachar N, et al. Unknown malcode detection using opcode representation[M]//Proceedings of the ist European conterence on Intelligence and Security Informatics. Springer-Verlag, 2008:204-215.

[12] 孔德光，谭小彬，奚宏生，等．提升多维特征检测迷惑恶意代码 [J]. 软件学报，2011, 22(3): 522-533.

[13]Skoudis E, Zeltser L. Malware: Fighting malicious code[M]. Upper Saddle Rivor, USA, Prentice Hall Professional, 2004.

[14]Szor P. The art of computer virus research and defense[M]. New York, USA, Pearson Education, 2005.

[15]Treadwell S, Zhou M. A heuristic approach for detection of obfuscated malware[C]//Intelligence and Security Informatics, 2009. ISI'09. IEEE

International Conference on. Sam Piego, USA IEEE, 2009: 291-299.

[16]Merkel R, Hoppe T, Kraetzer C, et al. Statistical detection of malicious PE-executables for fast offline analysis[C]//Communications and Multimedia Security. Springer-Verlag, 2010: 93-105.

[17]Belaoued M, Mazouzi S. Statistical Study of imported APIs by PE type malware[C]//Advanced Networking Distributed Systems and Applications (INDS), 2014 International Conference on. IEEE, 2014: 82-86.

[18]Belaoued M, Mazouzi S. An MCA Based Method for API Association Extraction for PE Malware Categorization [J]. International Journal of Information and Electronics Engineering, 2015, 5(3): 225.

[19]Yang J H, Ryu Y. Design and Development of a Command-line Tool for Portable Executable File Analysis and Malware Detection in IoT Devices[J]. International Journal of Security and Its Applications, 2015, 9(8): 127-136.

[20]Merkel R. Statistische Merkmale zur Anomaliedetektion in ausf ü hrbaren Dateien[D]. Ott0-dayvon-Guericke-University of Magdeburg, 2009.