计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240800062

基于指令流图特征的恶意文件的分类算法研究

邢昱阳　王宝会
北京航空航天大学软件学院　北京 100191
(15530312352@163.com)

摘　要　近年来,恶意代码愈加泛滥,数量和种类均呈快速增长趋势。因此,机器学习方法被广泛用于提升对恶意代码识别和分类的效率。聚焦恶意代码多分类任务,采用静态分析方法,结合反汇编、图构造、图论特征分析等技术,对恶意代码样本的原始文件进行特征提取。在传统的CFG特征和字节码特征的基础上,提出一种指令流程图(Instruction Flow Graph,IFG)特征。将IFG特征、CFG特征和字节码特征分别用于训练机器学习模型,并进行横向对比实验。从训练效果来看:相比CFG特征,采用IFG特征,模型精确率提高5%左右;相比字节码特征,采用IFG特征,模型精确率提高0.3%,模型训练时间缩短60%以上。

关键词:恶意代码;指令流程图;分类;机器学习
中图分类号　TP309

Research on Malware Classification Algorithm Based on Instruction Flow Graph

XING Yuyang and WANG Baohui
School of Software,Beihang University,Beijing 100191,China

Abstract　In recent years,malicious codes have become increasingly rampant,with both the quantity and types showing a rapid growth trend. Therefore,machine learning methods have been widely introduced to improve the efficiency of malicious code identification and classification. This paper focuses on the multi-classification task of malicious codes,adopts static analysis methods,and combines technologies such as disassembly,graph construction,as well as graph theories to extract features from the original files of malicious code samples. Based on the traditional CFG features and bytecode features,the IFG feature is proposed. The IFG feature,CFG feature,and bytecode feature are respectively used to train machine learning models for a horizontal comparison experiment. From the training effect:Compared with the CFG feature,using the IFG feature,the model'saccuracy rate increases by about 5%;compared with the bytecode feature,using the IFG feature,the model's accuracy rate increases by 0.3%,and the model's training time is shortened by more than 60%.

Keywords　Malicious code,Instruction flow graph,Classification,Machine learning

1　引言

恶意代码,是指一种针对计算机系统、网络和移动设备发起主动攻击的计算机程序。这种程序通常由黑客或病毒制作者编写,目的是破坏、控制或窃取信息和数据。近年来,全球恶意代码的数量逐年递增,根据AV-TEST统计,截至2023年年中,恶意代码总数已经接近11亿[1]。图1统计了历年恶意代码的数量。

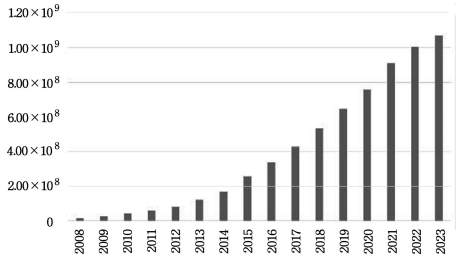


图1　历年恶意代码数量统计

Fig. 1　Statistics on the number of malicious codes in the past years

恶意代码种类繁多,常见的包括木马、蠕虫、病毒、后门、Rootkit、流氓软件、间谍软件、广告软件、僵尸网络等等。2022年,瑞星"云安全"系统共截获恶意代码样本总量达到7355万个,感染次数高达1.24亿次。其中,新增木马数量最多,占总体数量的62%。排名第二的是蠕虫,占总体数量的19%。后门、感染型病毒和灰色软件分别占总体数量的7%,7%和5%。由于实际功能、行为特征或传播方式存在差异,又会产生大量变种,形成不同的恶意代码家族。例如,AntiFW是一个木马家族名称,其主要功能是绕开防火墙执行恶意活动;Allaple是一个蠕虫家族代号,能通过从受感染计算机下载其他恶意文件,创建僵尸网络或窃取敏感信息。

恶意代码的数量和种类均呈快速增长趋势,传统的恶意代码分类技术存在明显弊端。如,基于签名的方法高度依赖签名数据库,难以检测新型恶意代码;基于启发式的方法采用动态分析手段,通过搭建虚拟机环境记录程序的行为(如文件操作、注册表操作、网络通信等)判断该程序是否存在恶意,可能占用大量计算资源且效率偏低。在此背景下,研究人员广泛引入机器学习方法,利用大规模数据集和机器学习算法,使计算机系统能够更智能地检测和分类各种恶意代码变体。这种方法不仅提高了恶意代码检测的准确性和效率,还能快速

通信作者:王宝会(wangbh@buaa.edu.cn)

适应新型恶意代码，有效增强网络安全防御能力。

将机器学习方法应用于恶意代码分类任务时，首先要提取特征。恶意代码特征分为动态特征和静态特征。动态特征是指恶意代码在运行过程中表现的行为特征，需要构建特定的执行环境，包括硬件、操作系统和软件等，以确保恶意代码能够正常运行和被监测[2]。动态分析通常需要较长的时间来执行恶意代码并监测其行为，这会消耗大量的计算资源。

本文采用静态特征方法。静态特征通过处理恶意文件进行提取，具有安全性高、复杂度低、实时性强等优点。静态特征又可细分为简单静态特征和复杂静态特征。简单静态特征是指可以直接从恶意代码文件样本中获取的字符串特征，如文件结构特征、代码语法特征、代码结构特征、API 调用特征等。为了提高模型的准确率和鲁棒性，在简单静态特征的基础上采用特定方法进行加工，可以获得复杂静态特征。常用的加工方法包括：N-gram 方法、可视化技术、基于图论的方法等。

本文着重研究了基于图论方法对恶意代码文件样本提取特征并进行多分类的算法。主要有以下几点原因：1）基于图论进行特征提取的相关研究较少，具有一定创新性；2）大部分相关研究仅实现对恶意代码和良性代码的二分类，多分类任务较少；3）相比传统方法，该方法在模型训练速度和分类效果上有一定提升。

2 基于图论的恶意代码分类

基于图论的恶意代码特征提取方法运用图论概念和算法，从恶意代码各个层次的图结构中挖掘出能够辨识恶意代码的关键特征，为恶意代码检测、分类和防御提供支持。Jiang 等[3]采用基于函数调用图（Function Call Graph，FCG）的特征提取方法：首先，借助反汇编工具 IDA pro 将恶意代码样本转化为 FCG；接着，采用 node2vec 技术将函数调用图映射到低维特征空间，形成嵌入式向量；然后，引入其他特征与该嵌入式向量进行融合和降维；最终，形成 100 维度的特征向量，将其输入到深度神经网络中完成对恶意代码样本的分类。Yang 等[4]认为，采用全局 FCG 作为特征可能导致模型过于复杂，难以适应恶意代码实时检测的场景，因此提出了“API-pair”图。“API-pair”图是 2-gram、Marcov 链和图论的结合：将每一次 API 调用行为作为节点，采用最大熵原理计算节点权重，基于 API 调用行为之间的递进关系构成连接节点的边。在此基础上，对“API-pair”进行聚类和筛选，并将处理后的样本输入到 LSTM 循环神经网络中完成分类。Abusnaia 等[5]提出一种基于控制流程图（Control Flow Graph，CFG）特征的恶意代码检测模型：先利用 Radare2 逆向工程框架获取良性样本和恶意样本的原始 CFG 并提取图论特征，包括中心性度量指标、节点数、边数和图形密度等；再将这些指标表示为特征向量输入到 CNN，DNN，RF 等模型中完成训练，解决了良性代码和恶意代码的二分类问题。

参考相关研究可知，字节码特征统计了二进制代码中字节码（0－255）的出现频次，是一种被广泛应用的简单静态特征，提取方法简单且性能稳定；CFG 特征是目前最常用的图论特征之一，配合多种机器学习或深度学习算法均可较好地完成恶意代码分类任务。在此基础上，本文创新性地提出一种基于指令流程图（IFG）的恶意代码多分类算法，并进行以下验证工作：

1）收集恶意代码样本并提取基准特征，包括字节码特征和 CFG 特征；

2）通过反汇编、图构造、图论特征分析等方法，对相同样本提取 IFG 特征；

3）基于机器学习和深度学习算法，分别将字节码特征、CFG 特征和 IFG 特征作为输入，训练恶意代码分类模型。

3 基于 IFG 的恶意代码多分类

本文提出的恶意代码多分类模型的框架如图 2 所示。首先，对恶意代码样本反汇编，将二进制文件转化为操作码指令集；接着，基于指令间的关系构造指令流程图；然后，基于图论对指令流程图提取 IFG 特征并形成特征向量；最后，将特征向量输入分类模型，训练并输出分类结果。

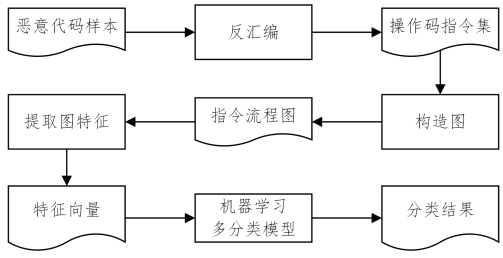


图 2 恶意代码多分类模型框架示意图

Fig. 2 Schematic diagram of malicious code multi-categorization model framework

3.1 获取操作码指令集

本文采集的恶意代码样本均为可执行文件，以二进制文件形式保存。如图 3 所示，可执行文件通常由 DOS 头、PE 文件头和节数据组成。节数据包含 text，data 和 rsrc 等节区，其中，text 节区保存了大部分主要执行代码。

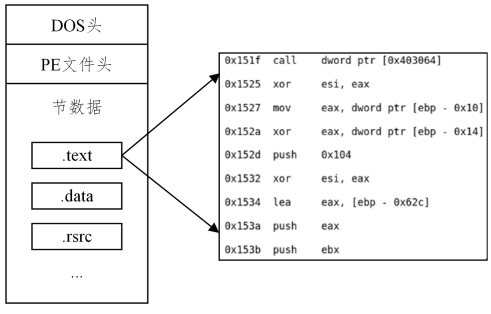


图 3 可执行文件结构示意图

Fig. 3 Schematic diagram of executable file structure

本文使用 Capstone 库对恶意代码样本进行反汇编，将二进制文件转化为可读性更强的标准汇编代码。提取 text 节区中的汇编代码，获得操作码指令集，每一行指令由 3 部分组成：地址、助记符和指令参数。

3.2 构造指令流程图

本文以每一行指令为一个节点，以指令间的递进、跳转和调用等关系为有向边，将操作码指令集转化为指令流程图。其中，递进关系表示指令的存储地址前后相邻，跳转和调用关系通过操作码的助记符判断。常见的跳转指令助记符有 JMP，JA，JB，JE，JZ 等；调用指令助记符为 CALL。构造指令流程图的伪代码如算法 1 所示。

算法 1 指令流程图构造算法

```
Input:assembly code
Output: G
Begin
  set G,Ad,Mn,Op
  for index,(Ad,Min,Op) in enumerate(assembly_code):
    if Mn.start_wih('J')or Mn==='CALL':
      target_address=Op
      G.add _edge(Ad,taget_address,jump_type = Mn)
    else:
      if index < length(assembly_code)-1:
        next_address = assembly_code[index+1][0]
        G.add_edge(Ad,next_address,jump_type = 'to')
End
```

算法 1 的输入为对恶意代码样本二进制文件转化所得的标准汇编代码，包含若干行指令。对每一行指令，提取地址 Ad、助记符 Mn 和指令参数 Op。如果助记符为跳转指令或调用指令，则从指令参数中提取目标指令的地址，在图中建立一条边，起点为当前指令地址，终点为目标指令地址，边的类型记为当前指令的助记符；如果助记符并非跳转指令或调用指令，则在图中建立一条边，起点为当前指令地址，终点为相邻的下一条指令地址，边的类型记为'to'。输出指令流程图 G 的可视化效果如图 4 所示。

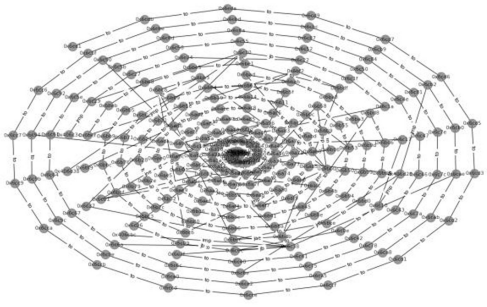


图 4 指令流程图可视化

Fig. 4 Instruction flow graph visualization

由恶意代码样本转化而来的指令流程图形态较为复杂，从中提取特征成为提高分类模型框架整体性能的关键。

3.3 基于图论提取特征

由恶意代码样本转化而来的指令流程图为有向图，本文主要提取以下几类特征。

1）节点出入度特征

节点出入度包含：出度（Out Degree），表示该节点指向其他节点的边的数量；入度（In Degree），表示其他节点指向该节点的边的数量。出度、入度高的节点在图中具有更高活跃度和更强影响力。

对出度和入度为 0 至 5 的节点数量进行统计，可以体现有向图中节点的整体活跃度和高活跃度节点的分布情况。通常来说，恶意代码会通过更多的跳转和调用操作来达到混淆的目的，因此，恶意代码的节点整体活跃度一般偏高。另外，不同家族恶意代码的混淆策略通常有所区别，可能导致高活跃度节点的分布呈现差异。

2）节点中心性特征

度中心性（Degree Centrality）的计算如式（1）所示。其中，$degree(v)$为节点出度和入度的总和。

$$C_d(v)=degree(v) \tag{1}$$

介数中心性（Betweenness Centrality）的计算如式（2）所示。其中，$\sigma(s,t)$ 表示从节点 s 到节点 t 的最短路径数量，$\sigma(s,t|v)$ 表示从节点 s 到节点 t 的最短路径中经过节点 v 的数量。

$$C_b(v)=\sum_{s,t\in V}\frac{\sigma(s,t|v)}{\sigma(s,t)} \tag{2}$$

紧密中心性（Closeness Centrality）的计算如式（3）所示。其中，$d(u,v)$表示节点 v 到节点 u 的最短路径长度。

$$C_c(v)=\frac{1}{\sum_{u\neq v}d(u,v)} \tag{3}$$

特征向量中心性（Eigenvector Centrality）的计算如式（4）所示。其中，$N(v)$表示与节点 v 直接相连的节点集合，λ 是最大特征值。

$$C_e(v)=\frac{1}{\lambda}\sum_{u\in N(v)}C_e(u) \tag{4}$$

卡兹中心性（Katz Centrality）的计算如式（5）所示，与特征向量中心性类似，但引入了一个衰减因子 β，考虑了不同距离的节点对中心性的贡献。

$$C_k(v)=\sum_{k=1}^{\infty}\beta^k\left(\sum_{u\in N(v)}C_k(u)\right) \tag{5}$$

谐波中心性（Harmonic Centrality）的计算如式（6）所示。

$$C_h(v)=\sum_{u\neq v}\frac{1}{d(u,v)} \tag{6}$$

这些中心性均可用于衡量节点在图中的重要性[6]。对某样本指令流程图中的所有节点计算以上 6 种中心性并绘制分布直方图，如图 5 所示，显然，中心性的分布并不均匀。再考虑到不同样本的节点数也存在差异，本文以每种中心性的平均值构建特征向量。

节点中心性特征可以进一步体现代码的整体结构特性。恶意代码通常会引入一些冗余代码段或数据，对其真实的恶意信息起到一定掩护作用。这类代码的信息一般较为分散，中心性偏低。

3）边特征

边特征用于表示指令流程图中节点之间的关系。本文以指令助记符区分边类型，统计边类型的出现频率以构造边特征向量。恶意代码中往往存在更多的跳转和调用操作。

4）结构特征

结构特征用于表示指令流程图的结构特性，具体包括：图的直径，即图中相距最远的两个节点之间的距离；图的密度，即图中边的数量与节点数量的比值，用于描述图中节点间关系的紧密程度；子图结构特征，表示图中的局部模式，如环形结构、三角形结构、四边形结构等。

结构特征能够直观地表示代码的执行路径、控制流和逻辑结构，更好地捕捉复杂的程序逻辑、分支、循环、跳转等结构化信息。

5）子图特征

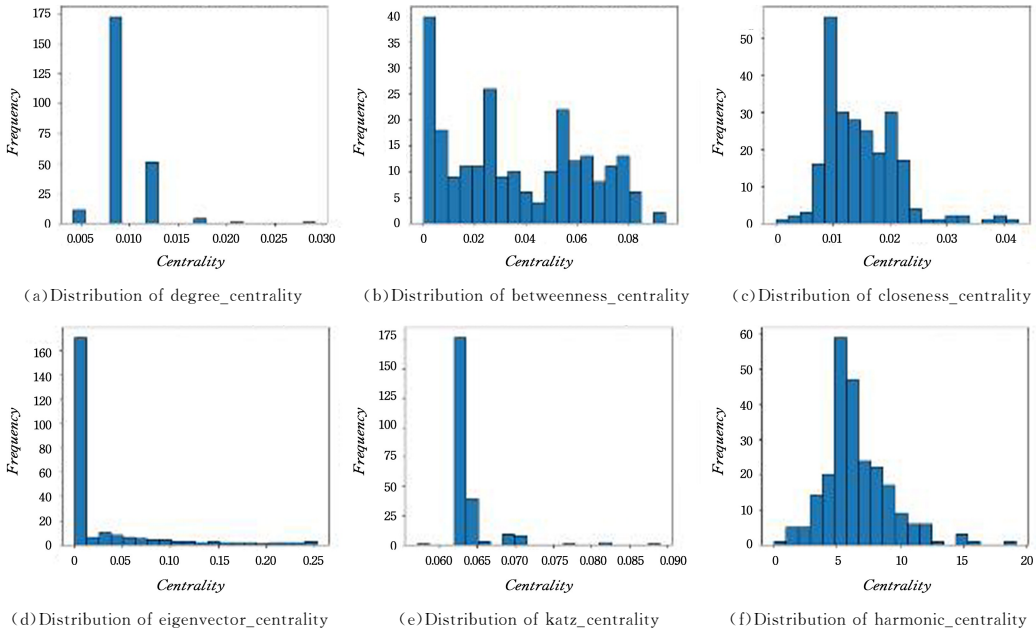
本文采用“Greedy”社群检测算法，将图剖分为若干个社群。该社群检测算法基于“Greedy”思想，在将节点划分入不同社群的过程中，迭代选择最大化模块度增益的方案，最终将图中所有的节点分配到若干个社群中。模块度的计算方法如式（7）所示。

$$M=\frac{1}{2m}\sum_{i,j}\left[A_{i,j}-\frac{k_ik_j}{2m}\right]\delta(c_i,c_j) \tag{7}$$

其中，M 表示模块度，模块度提升的本质是社群内连接度的

最大化和社群间连接度的最小化；$A_{i,j}$表示图中节点的邻接矩阵；k_i和k_j表示节点i和节点j的度；m表示图中所有边的总数；$\delta(c_i, c_j)$是一个指示函数，节点i和节点j属于同一个社群时，该函数值为1，反之则为0。

本文选择最大社群构造子图，该子图可能包含最核心的代码，摒弃了一部分冗余或混淆的信息。对子图提取节点出入度、节点中心性、边、结构等特征，进一步构造特征向量。



(a)Distribution of degree_centrality (b)Distribution of betweenness_centrality (c)Distribution of closeness_centrality

(d)Distribution of eigenvector_centrality (e)Distribution of katz_centrality (f)Distribution of harmonic_centrality

图5 某样本指令流程图的节点中心性分布

Fig. 5 Node centrality distribution of a sample instruction flow graph

3.4 恶意代码分类算法

目前主流的恶意代码分类算法模型包括：机器学习模型、深度学习模型和集成学习模型。这些算法模型各具优点和局限性。在实际应用中，需要综合考虑数据集、分析模式、特征提取策略等因素，选择合适的算法，以训练高性能的恶意代码分类模型。本文依次采用4种分类算法，分别训练和验证，并横向对比综合性能。这4种算法分别为：K近邻(K-Nearest Neighbors，KNN)、随机森林(Random Forest，RF)、XGboostt(eXtreme Gradient Boosting)和一维卷积神经网络。其中，K近邻是最基础的机器学习算法之一，其训练速度通常较快；大量相关研究证明，随机森林和XGboost对于恶意代码分类任务，是较为实用且高效的机器学习算法；深度学习模型也被广泛应用于恶意代码分类中。本文仅采用较为基础的一维卷积神经网络，主要原因在于：一方面，深度学习模型的训练速度通常较慢；另一方面，本文研究重点在于特征提取方法，分类算法方面的进一步研究暂不展开。

1)K近邻

K近邻算法是一种基本的监督学习方法，它通过测量待分类样本与训练数据中最接近的K个邻居之间的距离，来决定该样本的类别或属性值。KNN算法简单而直观，适用于多个领域，但需要选择适当的K值和距离度量方法，同时在处理大规模数据集时可能面临计算复杂度的挑战。在恶意代码多分类问题中，KNN算法的K值取为家族类别的数量，距离度量方法则采用欧氏距离。欧氏距离的计算方法如式(8)所示，其中n表示特征的数量，x_i和y_i分别表示样本x和样本y在第i个特征上的取值。

$$D(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2} \tag{8}$$

2)随机森林

随机森林是一种基于决策树的集成学习方法，该算法以Bagging策略为基础，通过自助采样创建多个不同的训练数据集。自助采样是一种有放回抽样方法，每次从原始数据集中随机选择一个样本，并将其放回训练集，从而可能在同一个训练集中多次选择相同的样本，也可能在某次采样中未选择某些样本(即袋外样本)。在构建决策树的过程中，对于每个节点的分裂，随机选择一部分特征以增强模型的多样性。对于恶意代码多分类问题，首先利用每个独立决策树对样本进行分类，得到多个预测结果，接着采用投票机制，选择得票最多的类别作为随机森林的最终预测结果。

总体而言，随机森林算法通过多个决策树的集成，有效提高了多分类结果的准确性，同时利用Bagging策略和随机特征选择方法有效抑制过拟合，具备对高维和大规模数据集的鲁棒性[7]。

3)XGBoost

XGBoos算法与随机森林算法存在相同点：第一，都是集成学习算法，通过组合多个弱学习器来构建一个强大的模型；第二，基本学习器都是决策树；第三，具有鲁棒性，能容忍一定的噪声和异常值。然而，二者的基本原理存在明显差异：首先，XGBoost采用梯度提升技术，通过在每一轮迭代中调整模型参数，最小化损失函数，从而减小误差，因此XGBoost能够更好地适应复杂的数据结构，对于高度非线性的问题表现更为出色；其次，XGBoost引入了正则化策略，包括L1(Lasso)和L2(Ridge)正则化，用于控制模型的复杂度，防止过拟合[8]。相较之下，随机森林通常需要抑制树的数量，以控制模型的复杂度。

4)卷积神经网络

卷积神经网络(Convolutional Neural Network，CNN)在

计算机视觉领域被广泛应用，利用卷积、池化、非线性激活函数和反向传播等技术实现对图像特征的高效提取。基于从恶意代码文件中抽取的特征向量或特征图像，卷积神经网络能高效识别恶意代码的模式和特征，进而实现恶意代码的识别和分类[9-11]。本文定义了一种为一维输入序列的多类分类任务而设计的 CNN 架构，如图 6 所示。

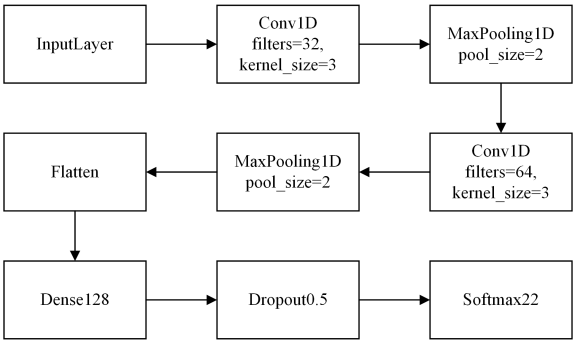


图 6　一维卷积神经网络架构示意图

Fig. 6　Schematic diagram of 1D convolutional neural network architecture

该模型结构包含两个使用 ReLU 激活函数的 Conv1D 层，通过 MaxPooling 进行降采样，然后连接一个具有 128 个单元的全连接层（ReLU 激活）。最后，通过 Softmax 层实现多分类输出，使用 Adam 优化器提高模型性能和泛化能力。该模型适用于恶意代码样本的分类任务。

4　实验和分析

4.1　数据集

本文实验使用从 VirusShare 获取的 9 109 个恶意代码样本，这些恶意代码样本以“卡巴斯基家族划分方式”进行命名，如图 7 所示，可通过文件名直接获取样本的类型和家族信息。

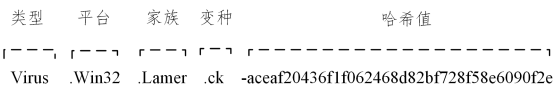


图 7　VirusShare 恶意代码样本命名方式

Fig. 7　VirusShare malicious code sample naming scheme

本文选取其中 22 个家族，对每个家族抽取 20%的样本形成测试集，剩余 80%的样本作为训练集。

4.2　实验环境

本文实验环境的主操作系统为 Windows 10，同时配置 Ubuntu 虚拟机环境，用于存放及处理恶意代码样本，具体配置参数如表 1 所列。

表 1　实验环境配置参数

Table 1　Configuration parameters of experimental environment

	配置参数
操作系统	Windows 10
虚拟机环境	Ubuntu-22.04.2
CPU	Intel(R) Core(TM) i7-10700
内存	64 GB
硬盘	1 TB
显卡	NVIDIA GeForce RTX 2080Ti
开发语言	Python 3.9.13

4.3　评价标准

本文采用精确率（Precision）、召回率（Recall）和 F1 值 3 种评价指标衡量模型对恶意代码样本的分类效果。

$$Precision=\frac{TP}{TP+FP} \tag{9}$$

$$Recall=\frac{TP}{TP+FN} \tag{10}$$

$$F1=\frac{2\times Precision\times Recall}{Precision+Recall} \tag{11}$$

其中，TP 为正样本被正确地识别为正样本的数量，TN 为负样本被正确地识别为负样本的数量，FP 为负样本被错误地识别为正样本的数量，FN 为正样本被错误地识别为负样本的数量。

精确率表示被预测为某一家族的恶意代码样本被正确预测的比例；召回率表示实际为某一家族的恶意代码样本被正确预测的比例；F1 值是精确率和召回率的调和平均值。

4.4　实验结果分析

为了验证基于指令流程图的恶意代码多分类算法的有效性，本文分别基于字节码特征、CFG 特征、IFG 特征，使用 KNN，RF，XGBoost 和 CNN 等模型进行恶意代码多分类。记录不同“特征类型＋分类模型”组合实现的精确率、召回率和训练时间，如表 2 所列。

表 2　结果数据

Table 2　Result data

特征类型＋分类模型	时间/s	*Precision*/%	*Recall*/%	*F1*/%
字节码＋KNN	3.9	84.0	89.4	86.6
字节码＋RF	31.3	93.1	93.0	93.0
字节码＋XGBoost	106.2	93.9	93.3	93.6
字节码＋CNN	139.6	93.3	93.3	93.3
CFG＋KNN	0.8	81.1	81.4	81.2
CFG＋RF	10.8	88.0	85.3	86.6
CFG＋XGBoost	23.3	87.6	84.4	86.0
CFG＋CNN	134.3	88.9	90.8	89.8
IFG＋KNN	1.5	89.2	90.8	90.0
IFG＋RF	12.7	94.2	95.6	94.9
IFG＋XGBoost	22.5	94.2	94.6	94.4
IFG＋CNN	137.5	93.3	94.2	93.7

从实验数据可知，本文研究的 IFG 特征，在采用 RF 模型时得到最好的分类效果，精确率为 94.2%，召回率为 95.6%，F1 值为 94.9%；相比之下，字节码特征采用 XGBoost 模型时分类效果最佳，精确率为 93.9%，召回率为 93.3%，F1 值为 93.6%；CFG 特征采用 CNN 模型时分类效果最佳，精确率为 88.9%，召回率为 90.8%，F1 值为 89.8%。IFG 特征在准确性方面有所提升，主要原因在于：仅对恶意样本保存在 text 节区的主要执行代码进行提取和处理，规避了其他节区可能存在的冗余、干扰或迷惑性数据。另外，在使用 KNN，RF 和 XGBoost 等机器学习分类模型的情况下，IFG 特征的训练时间与 CFG 特征近似，相比字节码特征明显缩短。综上，相比 CFG 特征，IFG 特征的 F1 值提高 5%左右；相比字节码特征，IFG 特征的 F1 值提高 0.3%左右，训练时间缩短 60%以上。

结束语　本文提出了基于指令流程图的恶意代码多分类模型。首先，利用反汇编技术处理恶意代码样本，获取保存于 text 节区的主要执行代码及操作码指令集；接着，将操作码指令集转化为指令流程图，并基于图论提取节点特征、边特征、结构特征和子图特征，形成 IFG 特征向量；然后，将 IFG 特征向量输入 KNN，RF，XGBoost 和 CNN 等模型中，实现对包含

22个恶意代码家族样本数据集的多分类;最后,与字节码特征和CFG特征进行横向对比,实验数据表明,IFG特征在准确性及训练速度方面,实现了综合性能的提升。

未来将在本文研究的基础上,进一步研究恶意文件的静态特性,优化IFG特征,并尝试结合图表征学习,提升分类模型的综合性能。

参 考 文 献

[1] AV-TEST: The Independent IT-Security Institute[EB/OL]. https://www.av-test.org/en/statistics/malware.

[2] BHATIA T, KAUSHAL R. Malware detection in android based on dynamic analysis[C]// 2017 International Conference on Cyber Security and Protection of Digital Services(Cyber Security). IEEE, 2017.

[3] JIANG H, TURKI T, WANG J T L. DLGraph: Malware detection using deep learning and graph embedding[C]// 2018 17th IEEE International Conference on Machine Learning and Applications(ICMLA). IEEE, 2018: 1029-1033.

[4] YANG S, LI S, CHEN W, et al. A real-time and adaptive-learning malware detection method based on API-pair graph[J]. IEEE Access, 2020, 8: 208120-208135.

[5] ABUSNAINA A, ABUHAMAD M, ALASMARY H, et al. Dl-fhmc: Deep learning-based fine-grained hierarchical learning approach for robust malware classification[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 19(5): 3432-3447.

[6] AGUIRRE J, PAPO D, BULDÚ J M. Successful strategies for competing networks[J]. Nature Physics, 2013, 9(4): 230-234.

[7] GOYAL M, KUMAR R. Machine Learning for Malware Detection on Balanced and Imbalanced Datasets[C]// 2020 International Conference on Decision Aid Sciences and Application(DASA). IEEE, 2020: 867-871.

[8] KONG Z, XUE J, WANG Y, et al. MalFSM: Feature Subset Selection Method for Malware Family Classification[J]. Chinese Journal of Electronics, 2023, 32(1): 26-38.

[9] WU Z, ZHANG J, KOU L. A Model for Malware Detection Method based on API call Sequence Clustering[C]// 2022 9th International Conference on Dependable Systems and Their Applications(DSA). IEEE, 2022: 1049-1050.

[10] SRIRAM S, VINAYAKUMAR R, SOWMYA V, et al. Multi-scale learning based malware variant detection using spatial pyramid pooling network[C]// IEEE INFOCOM 2020—IEEE Conference on Computer Communications Workshops(INFOCOM WKSHPS). IEEE, 2020: 740-745.

[11] ALAM M, AKRAM A, SAEED T, et al. DeepMalware: A Deep Learning based Malware Images Classification[C]// 2021 International Conference on Cyber Warfare and Security(ICCWS). IEEE, 2021: 93-99.

XING Yuyang, born in 1992, postgraduate. His main research interests include network security, big data and artificial intelligence.

WANG Baohui, born in 1973, professor, master supervisor. His main research interests include network security, big data and artificial intelligence.