基于图像可视化的恶意软件分类技术综述

钱丽萍 [1], 王大伟 [2]

[1] 北京建筑大学 电气与信息工程学院 北京 中国 100044
[2] 国家计算机网络应急技术处理协调中心 北京 中国 100029

摘要 恶意软件在研制中日益呈现出规模化、家族化、自动化趋势, 并普遍采用加密和混淆技术去对抗检测, 既带来恶意软件数量快速增长, 其内在隐含的特征又为深度学习检测提供了潜在可能, 因此主流检测和分类方法已从基于人工特征匹配转向基于机器学习及深度学习自动挖掘。恶意软件分类模型的性能往往取决于人工专家所挖掘的分类特征的质效。将恶意软件映射为图像, 既有助于缓解人工特征工程面临的专业知识匮乏, 亦可自然借鉴图像分类领域的最先进成果, 基于图像可视化的恶意软件分类技术成为一个重要的研究方向。本综述对基于图像可视化的恶意软件分类技术进行总结, 重点研究了恶意软件图像的生成模式, 包括图像大小、灰度或彩色通道选择、像素位置映射以及像素值计算等, 对比分析不同图像表征和特征抽取方法以及分类性能。由不同文献试验结果, 可以发现上述因素对恶意软件的分类性能均有影响。然后总结基于图像可视化的恶意软件分类方法的优势, 提出其面临的主要问题和挑战, 其中优势包括有利于缓解对专家知识的强依赖、更适用于恶意软件变种的检测及可以借鉴图像处理领域的系列研究成果; 问题和挑战包括恶意载荷定位困难、模型适用性、结果可解释性以及高质量标注数据稀缺问题。面对这些问题及挑战, 展望了未来几个可行且重要的研究方向, 包括: 基于认知的深度学习模型、恶意软件领域知识图谱、恶意软件样本对抗增强以及评估基准与高质量数据集。

关键词 恶意软件分类; 恶意软件可视化; 特征工程; *x*-plot; 数据集; 数据增强
中图法分类号 TP391 DOI 号 10.19363/J.cnki.cn10-1380/tn.2024.09.07

A Survey on Image Visualization Approaches-based Malware Classification Techniques

QIAN Liping[1], WANG Dawei[2]

[1] College of Electrical & Information Engineering, Beijing University of Civil Engineering and Architecture, Beijing 100044, China
[2] National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China

Abstract As matters stand, malware makers have been developing malwares at the scale of an automated and family-run manner and generally utilizing code encryption and obfuscation techniques to combat the malware detection systems. The trend is profoundly mixed: the number of malwares increases rapidly and the potentiality of mining implicit features of malware variants with deep learning. The mainstream approach for malware detection or classification has shifted from artificial feature matching to deep mining. Performance of malware classification models often depends on the quality and efficiency of the artificial feature engineering. Mapping malware into image not only tends to alleviate the lack of professional knowledge in artificial feature engineering, but naturally borrows the achievements in the field of image processing. Image visualization approaches-based malware classification techniques has become an attractive research direction. The survey summarizes the research progress in image visualization approaches-based malicious executable classification techniques and focuses on the methods for generating image from executable file. We systematically summarize the methods for generating the image from a malware binary file, including image size setting, gray or color channels choice, pixel coordinates projection and pixel value computation, and gives comparative analysis on techniques of feature representation and extraction for visualized malware. Results from the listed literatures shows that the above factors all have an impact on the performance of malware classification. We also conclude its advantages and main difficulties and challenges encountered, which includes advantages of relieving strong reliance on expert knowledge, more applicable for detecting malware variants and drawing on series of research achievements in the field of image processing, and difficulties and challenges in locating malicious payloads, model applicability, interpretability and lack of high-quality labeled data. We then present several interesting directions for future research, such as cognitive DL models, malware knowledge graph, adversarial malware data augmentation, evaluation benchmark and high-quality dataset.

Key words malware classification; malware visualization; feature engineering; *x*-plot; dataset; data augmentation

通讯作者: 钱丽萍, 博士, 副教授, Email: qianliping@bucea.edu.cn。
本课题得到国家重点研发计划项目(No. 2022YFC3321101, No. 2022YFB3103705); 国家自然科学基金项目(No. 61571144)资助。
收稿日期: 2023-07-09; 修改日期: 2023-12-07; 定稿日期: 2024-04-24

1 简介

当前, 互联网作为数字时代的关键基础设施, 承载着巨大利益, 已成为各种利益群体竞争博弈的重要舞台。根据洛克希德·马丁公司(Lockheed Martin)提出的网络安全威胁的杀伤链模型(Cyber-Kill-Chain)[1], 攻击者在“载荷投递”和“命令与控制”阶段大概率会以恶意软件的方式将攻击能力投递到目标系统中, 即各类攻击活动最重要的启动方式就是通过恶意软件发起攻击并控制目标系统。根据CNCERT《2021 年上半年我国互联网网络安全监测数据分析报告》[2]: 2021 年上半年共捕获恶意程序样本约 2307 万个, 涉及家族约 20.8 万余个, 境内计有 3048 万个 IP 地址受攻击, 境外约 4.9 万个恶意程序控制服务器控制了境内约 410 万台主机。我国因此成为受网络攻击影响最大的国家, 但反过来又因受控主机被用于攻击跳板而备受境外敌手恣意诬陷。因此第一时间检出恶意软件攻击并及时处置, 对保障网络系统安全运行, 缓解关键信息基础设施遭受 DDoS 攻击及防范重要数据泄露等网络安全风险具有重大现实意义。

之前, 恶意软件检测普遍采用静态特征或动态特征进行指纹匹配和统计学习, 这些特征由专家设计或机器学习获得。随着恶意软件普遍使用打包、加密、多态、变形和混淆等逃避技术[3], 指纹和统计特征日益模糊, 这类检测技术面临功效衰退和时效滞后问题。另一方面, 恶意软件日趋复杂, 但其编制展现出新特点: 一是创建恶意软件的克隆版本、借用已知代码及利用开源软件代码(OSS, Open-Source Software), 远比新写程序简单。一般恶意软件编制所需知识明显下降, 盛行复制-粘贴, 基于公开漏洞利用和第三方库, 普遍使用开源程序或自动化工具创建[4]。二是高价值恶意软件趋于武器化, 以家族模式迭代。三是规模生成恶意软件开始成为一种互联网服务, 表现为付费定制模式交付[5]。这些新特点既带来恶意软件数量快速增长, 其内在隐含的特征又为深度学习检测提供了潜在可能。

深度学习在计算机视觉领域得到了广泛实践和成功。恶意软件因开发智能化和家族同源性, 使得同一家族的恶意软件映射生成的图像展现出明显的视觉相似性。研究人员普遍认为恶意软件可视化方法有助于有效缓解代码逆向和特征工程所依赖的专业知识匮乏, 应对原始二进制文件中恶意载荷模糊化、代码混淆以及潜在未知变种所带来的挑战, 更可以自然地借用计算机视觉领域的现有成果。本文对相关文献进行综合分析, 对采用字节、指令码等各类特征(以 x 代表)去映射图像的方式, 以 x-plot 加以概括, 并针对不同图像映射方法、不同图像特征等对可视化恶意软件分类性能的影响开展比较研究, 提出问题和挑战, 梳理了未来可能的研究方向。

2 相关工作

恶意软件与良性程序一样, 其最基本的形式就是符合特定格式规范的可执行文件。本文中所称恶意软件为内嵌恶意载荷的可执行代码, 即恶意可执行文件。其格式多样, 但目前最为广泛应用的是 Windows 平台的 PE 格式(Portable Executable)程序和 Android 平台的 APK 应用程序, 它们也是 95%以上文献所研究的目标。像 PDF 文件、Word 文件等文件格式, 并非作为可执行文件设计使用, 但因其支持宏指令而可以在内存中执行某些功能, 有时亦被用于携带某些恶意载荷。虽然多数研究重点关注 Windows PE 格式和 Android APK 格式恶意软件分类, 但相关技术亦可适配多种指令架构和文件格式。

2.1 恶意软件分类总体框架

总体而言, 可执行程序的格式规范仅定义了文件的结构特征, 无法区分一个程序是否恶意。当前复制化、自动化、家族化的恶意软件生产模式, 又使得恶意软件变种之间仍然存在深度模糊的相似特征。恶意软件分类就可以综合利用有效的区分特征、合理的分类模型去实现恶意软件家族分类。综观现有的恶意软件分类研究, 如图 1 所示, 总体框架大致由三部分构成, 即: 恶意软件表征、恶意软件分析、分类评估, 三部分并非严格切分, 体系上往往融合设计。

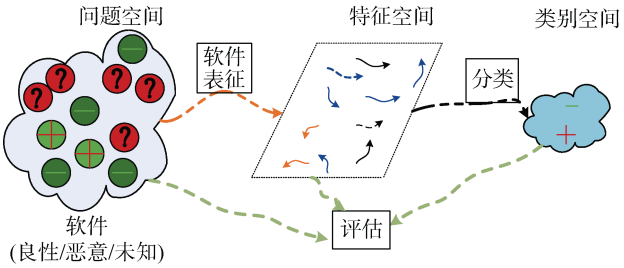

图 1 恶意软件分类总体框架

Figure 1 Framework of malware classification system

2.1.1 恶意软件表征

软件表征又可进一步分为预表征、特征抽取和特征选择三部分。同样这三部分并非严格切分, 如预表征与特征抽取、特征抽取与特征选择往往可以融合设计。预表征即以何种视图以计算机可理解的形式去表示一个可执行程序, 是为了便利后续的特征

抽取而进行的预处理操作, 因而与恶意软件分析欲使用的特征密切相关。例如: 如果要采用字节级指纹或字节级 *n*-gram 统计特征作为分析特征, 则直接将可执行程序视作一个二进制流; 如果要采用指令码统计特征或指令码序列作为分析特征, 则需要将可执行程序反汇编或反编译后抽取出所有指令码; 如果要采用函数或 API 的隐现作为分析特征, 则需要将可执行程序映射成一个函数或 API 调用隐现状态表; 如果要采用 API 调用序列作为分析特征, 则需要通过在仿真环境中执行该程序来将其映射到一组动态 API 调用序列; 如果要采用灰度图像统计特征作为分析特征, 则需要将可执行程序映射为一幅灰度图像等等。典型特征与预表征视图的关系如表 1 所示。

表 1 典型检测特征与文件视图

Table 1 Detection features and the corresponding file view

特征	文件视图
字节指纹/*n*-gram、字符串、结构化熵	二进制流
指令码统计信息、指令码序列	反汇编或反编译程序抽取的指令码序列
API 调用序列	仿真执行得到的所有 API 调用
API 调用隐现	API 调用隐现状态表
图像特征	字节值映射为像素值得到的灰度或彩色图像
Android App 的权限、activity、service、broadcast receiver、intent-filter、…	AndroidManifest.xml
PE 段表统计特征	PE 文件结构
控制流图、函数调用图	代码基块关系、函数调用关系

特征抽取是从预处理后的文件中抽取出所需的特征, 可以是结构特征、内容特征、元素列表特征、序列特征及统计特征等。特征选择则从抽取出的众多特征中挑选一部分重要特征, 原则是所选出的特征总维数既显著少于原始特征集, 从而有效降低分类或检测的计算负荷, 避免特征维数灾难, 同时仅利用这部分特征所实现的分类性能与原始特征集相比具有可以接受的损失。恶意软件分析中常用的特征选择策略包括主成分分析、线性判别分析、信息增益、递归特征消除等。

对于深度学习模型来说, 由于恶意软件属于几乎无法用任何显式规则描述语义的离散对象样本, 为解决特征值离散表示带来的稀疏和缺乏语义距离问题, 常将其以向量<0, 0, ..., 0, 1, 0, ...,0>指示某个特征是否出现的独热(One-hot)表示方式, 转换为隐含特征语义的稠密向量形式的嵌入(Embedding)表示方式, 如图 2 所示。例如, MalConv 不是将 PE 文件字节中每个字节直接输入卷积神经网络分类模型, 而是在首个卷积层前增加了一个嵌入层, 将每个字节映射为一个 8 维嵌入, 从而可以利用输入字节与其上下文中其他字节值之间隐含的“内在”紧密性[6]。Scorpion 从给定文件抽取文件、目录、机器、API、DLL 等内容特征, 采用结构化 HIN(Heterogeneous Information Network)去表示文件, 提出一种基于元图的方法去刻画文件之间的关系, 再用 HIN 嵌入模型 Metagraph2vec 去学习 HIN 中结点的低维表示, 从而能够保留不同类型结点间的语义和结构关系[7]。

2.1.2 恶意软件分类

恶意软件分类是指根据恶意软件的某些特征, 利用某种算法或模型, 自动将其分成不同种类或家族的过程。对于恶意软件和良性程序二类分类, 或者是特定恶意软件对象的鉴别, 一般使用检测、识别等概念。恶意软件的分类模型主要可以归为四类: 基于指纹匹配、基于特征统计、基于传统机器学习和基

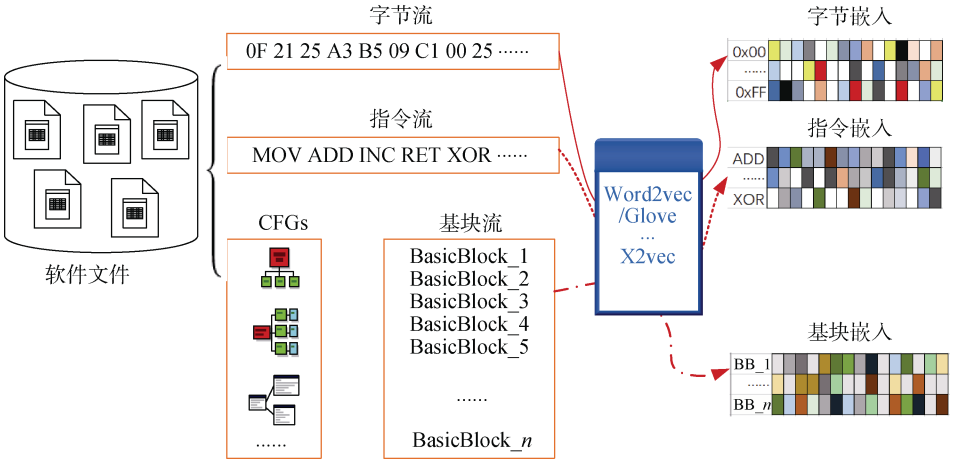

图 2 恶意软件离散特征的嵌入表示

Figure 2 Embeddings of the discrete features of malware

于深度学习。总体上, 这几类方法的技术复杂度渐次提升、对足以表征恶意软件家族的代表性数据要求渐次提高、结果可解释性渐次降低。

2.1.3 分类评估方法

与高性能计算领域具有 Linpack 基准[8]、系统性能测试领域具有 TPC 系列[9]和 SPEC 基准[10]不同, 恶意软件分类性能的评估缺乏公开时新的测试数据集和通用的测试评估方法。学术文献中最常使用的公开数据集包括 Microsoft Kaggle 2015 Windows 恶意软件数据集(后称 BIG 2015 数据集)、Nataraj 2011 Malimg 恶意软件图像数据集(后称 Malimg 数据集)和 Drebin 2014 安卓恶意软件集(后称 Drebin 数据集)等。其中: BIG 2015 数据集是微软为 Kaggle 2015 恶意软件分类挑战赛提供的 PE 文件数据集, 包含 9 个恶意软件家族 2 万多个恶意软件样本, 每个恶意软件以二进制码文件和反汇编程序两种格式提供, 为避免恶意利用, 二进制文件已将可执行文件的 PE 头部去除而不可执行。Malimg 数据集是 Nataraj 等 2011 年收集的 PE 文件恶意软件并以图像格式表示提供的数据集, 采用 Byte-Plot 方法生成, 其生成算法可参见 https://sarvamblog. blogspot.com/2014/08/ super-vised-classifica tion-with-k-fold.html, 包含 25 个家族的 9342 个恶意软件样本映射生成的灰度图像文件。Drebin 数据集是 2010—2012 年间收集的 Android 恶意软件数据集, 包含 179 个不同家族的 5560 个恶意软件应用程序。总体来说, 这些公开可用恶意软件数据集早以过时, 但仍广泛作为基准数据集使用。

恶意软件分类结果评估通常包含分类效果评估、计算性能评估与模型评估。与网络安全领域其他分类问题类似, 在评估分类效果时, 可以使用的指标包括: 准确率(Accuracy)和总体准确率、精确率(Precision)、召回率(Recall)、F1-Measure 综合指标等, 在此基础上, 还可引入 ROC(Receiver operating Characteristics)曲线和 AUC 面积。ROC 曲线描述了真阳率(TPR)和假阳率(FPR)的相对变化关系, 可用于直观地展示一种方法在不同参数时、或对比不同方法的分类准确率。ROC 曲线所包围的面积即为 AUC 面积, AUC 面积越大, 则分类模型在 TPR 和 FPR 上的均衡效果越好, 分类结果的可信度越高。计算性能评估主要分析分类模型的时空复杂性。模型评估则评估模型的鲁棒性、结果可解释性等。

2.2 恶意软件分类方法

对于恶意软件家族的简单变种, 可以通过结构相似性进行初步判定。而能够区分一个程序是否恶意的最基本特征只有两种: 一是是否包含恶意载荷, 二是是否执行恶意行为, 分别对应于恶意软件的静态特征和动态特征。基于静态特征实施恶意软件分类的方法称为静态分类方法, 基于动态特征实施恶意软件分类的方法称为动态分类方法, 同时使用静态特征和动态特征实施恶意软件分类的方法称为融合分类方法。鉴于越来越多的研究使用多类特征进行恶意软件分类, 我们将融合分类方法进一步归入多模态分类方法。

2.2.1 静态分类方法

静态分类方法所使用的静态特征是广义的, 包含指纹、字节 *n*-gram、指令码序列、控制流图、资源需求及其统计特征、熵特征等, 一般通过分析可执行代码提取, 或者将可执行代码反汇编或反编译成源代码后再提取。静态分类方法典型地又包含基于字节 *n*-gram 的方法、基于指令特征的方法、基于对象列表的方法、基于语义图的方法。

字节 *n*-gram 方法源于指纹法。Zhang J.等人组合使用基于 bi-gram 指令码嵌入表示特征和基于 API 频率向量嵌入表示特征, 在私有数据集上获得 95.10%的分类准确率[11]。Zhang G.等人采用 bi-gram 方法提取字节序列作为静态特征, 在 BIG 2015 数据集上, XGBoost 达到 98.43%的分类准确率, 优于 *k*-最近邻、逻辑回归和随机森林[12]。

基于指令特征的方法首先用反编译或反汇编工具, 将可执行程序反编译或反汇编为汇编程序或其他可读代码程序, 从中提取出指令用于分类。GEMAL 将指令视作单词, 函数视作句子, 基于 NLP 方法自动抽取语义特征, 使用基于注意力机制的图嵌入网络, 将二进制文件的结构特征及其汇编文件的语义特征组合成嵌入向量用于恶意软件分类[13]。

基于对象列表的方法利用恶意软件的 API 调用列表、DLL 装入列表、可打印串列表等的呈现或统计信息对恶意软件进行分类。Rafiqul 等人用函数长度频度和可打印串静态特征与 API 调用名及参数动态特征形成合成向量, 用于恶意软件分类[14]。

基于语义图的分类方法利用恶意软件中隐藏的语义结构信息进行分类。常用的语义图包括: 控制流图(Control Flow Graph, CFG)、函数调用图(Function Call Graph, FCG)、数据流图、系统依赖图等。通常一个程序由多个函数组成, 每个函数又包含多个由条件转移/跳转分割而成的顺序执行块, 前者对应于 FCG, 后者对应于 CFG。同一家族的恶意软件往往具有恶意功能和程序结构的相似性, 体现为 FCG 和 CFG 的相似性。Ahmed 等人设计了基于恶意软件 CFG 的图嵌入增强方法(Graph Embedding and Aug-

mentation, GEA), 构造 IoT 对抗软件[15]。BinAuthor 研究恶意软件作者关系分析和函数指纹技术, 用语义集成图表示二进制代码[16]。MAGIC 扩展了一种特殊类型的基于图内核的深度图卷积神经网络(DGCNN), 分类基于 CFG 表示的恶意软件[17]。Wang 等人将程序转换为由一个函数调用图和多个控制流图组成的层次图结构, 用 BERT 和图注意网络实现了一个语义表示更好、泛化能力更强的恶意软件分类模型[18]。Malgraph 用 FCG 和对应的 CFG 组成的分层图表示可执行文件, 然后使用基于图神经网络的端到端学习框架进行恶意软件检测, 评估结果表明 Malgraph 优于 MAGIC, EMBER 和 Malconv[19]。Sachin 等人[20]提出一种基于 CFG 类型的恶意软件机器学习分类算法, 取得与原始二进制文件相当的分类性能。DroindClone 将 Android 内生二进制码翻译成自定义的 MAIL (Malware Analysis Intermediate Language)中间代码格式, 再用 DroidClone 对 MAIL CFG 去检测 Android 应用程序中的代码克隆以协助检测恶意软件变种中的代码重用或家族克隆。基于 CFG 或 FCG 的恶意软件分类可以利用其丰富的语义信息, 且可与功能强大的图神经网络相结合[21]。Yue 等人[22]采用 Flowdroid 工具构建反编译 smali 代码的 FCG, 提取 API 构建有向图, 利用 SDNE 图嵌入算法选择特征, 基于图注意力网络检测 Android 恶意软件。MBGINet 通过分析 FCG 和 CFG, 基于图分类网络鉴别挖矿恶意软件[23]。

字节 *n*-gram 的方法不仅适用于主机系统, 也是网络环境中旁路监测的典型配置, 需要事先确定恶意模式。基于指令特征的方法、基于对象列表的方法、基于语义图的分类方法首先需要将可执行文件进行反汇编或反编译, 再从中提出 CFG 或 FCG, 在大规模实时检测场景中的应用受到限制, 一般用于线下分析, 不适于实时部署。

2.2.2 动态分类方法

动态分类方法所使用的动态特征一般包含 API/函数调用序列、系统资源利用等及其统计特征。动态特征的提取需要实际运行程序并跟踪程序执行, 一般通过在沙箱等仿真环境中运行可执行程序抽取。动态分类方法典型地又包含基于 API 调用序列的方法、基于系统运行状态的方法。

基于 API 调用序列的方法采用程序执行所调用 API 序列特征分类恶意软件。Zhang Y.等人[24]利用 Cuckoo 沙箱 PE_imports 元素中抽取的前 1000 个 API 作为分类特征, 基于 ResNet18 和注意力机制实现恶意代码检测, 在包含有 47580 个恶意和良性样本的数据集上, 获得 97.76%的检测准确率。DACN 将恶意软件的 API 调用、DLL 装入和注册表操作三种动态特征分别映射到图像的 R、G 和 B 通道, 再基于胶囊网络捕获特征间的空间关系并用于分类, 实现 97.5%的分类准确率[25]。Li Ce 等人[26]使用深度学习模型捕获 API 序列的内在特征, 并用 Bi-LSTM 模块挖掘 API 之间的关系。Li Chen 等人[27]采用 LSTM 和 GRU, 基于 API 调用长序列对恶意软件变种进行分类。Angelo 等人[28]应用深度图卷积网络(DGCNN), 基于 API 调用序列学习恶意软件行为模式。

基于系统运行状态的方法通过监测程序运行时系统资源的占用或变化情况进行恶意软件分类。Chen J.等人[29]结合使用 Cuckoo 沙箱技术和 DynamoRIO 系统动态二进制插桩工具, 获取恶意代码样本 API 序列及其文件操作行为、网络行为、内存转储行为等信息, 使用 Word2vec 将 API 序列转换为数字向量, 采用双向门循环单元(BGRU)作为恶意代码检测模型, 在实验数据集上取得 96.38%的检测准确率。Malview 是一个交互式恶意软件可视化平台, 基于 Procmon 工具捕获的关于文件系统、注册表、网络、进程和内存映像等动态行为数据, 实现了基于签名模式和行为模式的分析匹配方法, 可用于分类恶意软件、定位恶意载荷、发现未知模式或探索时间依赖[30]。

动态分类方法一般需要在系统上驻留实时监测模块以获取程序运行时产生的特征, 对系统性能有一定影响。同时 API 动态组合序列数量巨大, 基于已有恶意序列模式检测导致召回率不高; 沙箱中运行环境相对简单, 恶意软件运行时取得的系统状态也不一定符合实际系统的复杂环境。因此动态方法在实网中一般会结合静态方法共同使用。

2.2.3 多模态分类方法

为充分挖掘利用恶意特征, 一些研究采用多模态方法, 采用连接模型或集成模型, 融合多类检测特征用于恶意软件检测。HYDRA 是一个基于多模态学习的恶意软件分类框架, 集成了人工和端到端的特征抽取组件, 包括基于 API 的前馈神经网络、基于指令码 *n*-gram 的 CNN、基于原始字节的 DeepConv, 再用全连接层将这些特征融合, 在 BIG 2015 数据集上的实验获得了 99.75%的分类准确率[31]。Yang 等人[32]选择恶意软件图像的 GIST 纹理特征、字节统计值和字节熵直方图特征、汇编程序指令码序列特征、字符串序列特征、PE 文件结构特征以及函数调用关系特征, 基于多特征集成学习, 在 Datacon 2020 数据集上准确率达到 95.99%。Hojjat 等人[33]研究基于静态分析特征的 ML 恶意软件分类器对于打包样本的

检测, 从 PE 文件结构、原始程序字节码及程序反汇编代码抽取包括 PE 头、PE 段、DLL import、API import、Rich 头、字节 *n*-gram、指令码 *n*-gram、字符串以及文件相关特征等静态特征, 证实从打包的可执行文件中所抽取的信息并不足以供 ML 模型泛化其对于未知打包器的知识, 从而不利于适应恶意软件作者经常更愿意使用定制打包程序而非现成打包软件的情况。Zhang J.等人[11]对指令码特征嵌入和 API 调用特征嵌入分别采用 CNN 和反向传播神经网络训练后, 使用 softmax 融合这些特征嵌入进行恶意软件家族检测。Ai-Hydra 融合使用 5 类静态特征(大小、计数、熵、熵点和装入地址表)、5 类动态特征(API、位置、互斥锁、网络和注册表), 并组合随机森林和 12 个隐藏层深度学习模型, 采用投票方法构建的恶意软件分类模型优于单一模型[34]。Rajasekhar 等人[35]提出一种基于融合特征集和 CNN 架构的 Windows PE 恶意软件检测框架, 并在包含静态特征(PE 段、PE Import)、动态特征(PE API)、图像特征(大小为 32×32 的灰度图像, 对应于 1024×1 的 1 维向量)的融合特征集上去评估深度学习模型架构和机器学习模型架构, 同时遴选出最优模型, 所提出模型使用融合特征集时可实现 97%的准确率, 在未知恶意软件数据集上亦表现出相似性能。

多模态分类方法可融合使用静态特征和动态特征, 分类效果和泛化性较好。

2.3 软件检测攻防对抗

为逃避检测, 恶意软件利用加密、多态、变形、混淆等技术及现成工具, 不断更新逃避技术。随着深度学习检测模型的广泛研究和应用, 恶意软件样本对抗生成成为攻防双方关注的一个热点, 防守者用其增强失衡的恶意样本数据, 攻击者意欲用对抗样本逃避深度检测模型。在数据增强方面, Ziyue 等人[36]将恶意软件进行代码可视化, 采用生成对抗网络进行数据增强, 生成更多的恶意代码样本。Zhiguo 等人[37]为解决恶意软件家族间样本不均衡问题, 使用 Cycle-GAN 模型进行数据增强。在将恶意软件映射成图像后, 更为简便的方法是直接采用旋转、缩放、剪切、反射等图像变换操作增强数据, 解决数据集中类别不均衡问题[38-39]。

在样本逃避方面, 生成恶意软件逃避样本的一个重要考虑是原始恶意功能的保全问题, 多采用限定扰动特征种类和限定修改位置的方式。其中文件尾部追加方式相当于将代码追加到一个程序运行不可达的区域, 不影响程序可执行性, 但会改变深度学习模型的输入模式[40-42]。Mahmoud 等人[43]对 21 种顶级反恶意软件产品针对 7 种混淆工具和 29 种混淆策略的有效性开展大规模评估, 发现代码混淆显著影响 Android 反恶意软件产品检测性能。MalGene 借用生物信息学使用的局部序列对齐算法, 鉴别用于逃避检测的调用事件和比较事件并构造逃避签名[44]。William 引入重要字节有目标遮挡去发现恶意软件中的重要字节区域, 测试了 9 类不影响恶意代码功能的微小修改[45]。DroidSieve 选出一组资源特征和一组句法特征, 发现权限在恶意软件检测中起到重要作用[46]。SecureDroid 从特征贡献及操纵成本两方面考虑其重要性[47]。Dazhi 等人面向对抗样本欺骗 Windows PE 恶意软件分类器这一问题, 研究显著性检测方法对于恶意软件分类的决策, 利用 CNN 分类器在显著性分布上存在的两类相似性, 提出显著性附加(Saliency Append)攻击方法, 单次查询即可产生对抗样本, 较其他基于附加的攻击方法有更高的攻击成功率。生成的样本可进一步用于分类器的对抗训练, 从而显著提升分类器的鲁棒性[48]。Riya 等人[49]基于 Malconv 架构, 利用快速梯度符号方法(Fast Gradient Sign Method, FGSM)生成对抗样本, 评估基于深度学习的恶意软件检测算法的弹性和可靠性。

2.4 相关综述

文献[3]简要回顾了恶意软件产业的发展。检测技术也从基于签名、基于启发式发展到基于云。认为智能型检测方法通常分为特征抽取和分类/聚类两个阶段, 且这些方法的性能严重依赖于所抽取的特征和分类/聚类方法, 因此对静态分析、动态分析、混合分析等特征抽取技术, 对 Filter、Wrapper 和 Embedded 三类特征选择技术, 以及对恶意软件检测中使用的典型分类/聚类技术进行了全面综述。文献[50]对多种基于 DL 的恶意软件分类模型进行了比较实验, 包括基于图像的 MLP 和 CNN 以及基于指令码的 CNN/RNN/LSTM/GRU 等。文献[51]对基于图像的恶意软件检测和分类技术进行了较为全面的比较分析, 将恶意软件二进制文件分别映射为基于色图的灰度图像、基于 3 色图的 RGB 图像和 Markov 图像, 采用 Gabor 过滤器抽取生成图像的特征, 用包含 CNN、VGG3 和 ResNet50 在内的 12 种神经网络架构, 在 BIG 2015 数据集、Malimg 数据集和自采良性程序数据集上构建两个数据集开展实验。文献[52]针对 PDF 和 Office 恶意文档的攻击技术和检测方法及其研究进展进行了分析总结。文献[53]对恶意软件动态分析逃避技术及防御方法进行了综述。文献[54]系统研究了当前多种恶意软件检测方法和业界标准的恶意软件检测引擎针对代码混淆的能力和性能变化。

这些检测器使用的代表性表征方法包括: 基于灰度图像、基于 CFG 邻接性、基于 CFG 算法、基于串、基于段、基于符号、基于十六进制转储(hexdump)以及基于特征融合等方法, 发现打包、剥离、填充等方法会显著降低大多数检测器的准确率, 基于十六进制转储的逻辑回归分类器最为鲁棒, 基线准确率稳定在 98.96%。文献[55]对 9 种基于学习的恶意软件分类方法进行了经验比较研究, 包括 4 种基于图像的方法、两种基于二进制码的方法和三种基于汇编码的方法, 在 4 个数据集上的实验表明没有哪个方法单一性能上显著超越其他方法。文献[56]的综述主要针对生成可执行的对抗性恶意软件样本, 对恶意软件分类器实施现实攻击。文献[57]从视觉分析的角度对恶意软件可视化系统进行了综述和分类, 重点是对用于单个恶意软件进行可视化分析、比较和摘要的工具和技术进行总结。文献[58]对 Windows PE 恶意软件检测器的对抗攻击和防御方法进行了分类综述, 重点基于白盒攻击和黑盒攻击的原理, 从特征空间和问题空间两个攻击角度阐述了攻击策略。

3 基于图像可视化的恶意软件分类方法

抛开无关恶意判定的文件结构特征, 应用程序本质上就是一个二进制文件, 即一个位流或字节流, 但因为其中每个字节所代表的角色不同, 可能属于某个指令码、操作数或字符串等, 亦可能仅用于对齐填充, 因此这个字节流并非一个语义熵均衡的字节序列, 对其进行语义解析必须结合文件格式规范、指令系统、恶意行为甚至编译系统的深度专业知识, 人工辅助特征工程的难度不断提升。在计算机视觉特别是图像处理领域, 深度学习因其对视觉特征的内生挖掘能力, 在大规模训练数据的加持下, 在多种图像处理任务中不断刷新最先进性能, 形成了一系列方法和预训练模型。如何将这些方法和模型引入恶意软件分析任务, 破解人工特征工程所需的精深专业知识和时间成本, 成为了一个探索性的选项。

基于图像可视化的恶意软件分类亦由软件表征、分类检测和结果评估三部分组成。实际上, 对于恶意软件分类, 最为关键的就是恶意软件预表征方法, 需要结合领域知识去表示恶意软件。相对来说, 特征抽取和选择、分类检测、效果评估则大多借鉴图像处理领域通用的算法、模型或方法。因此我们重点阐述恶意软件的图像预表征方法。

3.1 恶意软件的图像预表征

恶意软件的图像预表征需考虑两个问题: 一是如何为文件长度相差较大的恶意软件确定生成图像的大小? 二是如何确定像素点的坐标、像素值及图像的通道数?

3.1.1 图像大小设置

对于问题一, 一般采用“区间映射+缩放调整”这一经验方法, 即首先根据文件长度设置几个区间, 分别对应于不同宽度的初始图像矩阵, 长度则随文件长度动态变化, 如表 2 所示[59]。

表 2 初始图像宽度

Table 2 Initial width value of the visualized image

文件大小	参考图像宽度
<10KB	32
10KB~30KB	64
30KB~60KB	128
60KB~100KB	256
100KB~200KB	384
200KB~500KB	512
500KB~1000KB	768
>1000KB	1024

相关研究一般都遵循表 2 确定初始图像宽度。在生成初始图像矩阵后, 一般采用图像缩放方法将所有图像规格化到相同尺寸, 该尺寸与后续选择的深度模型架构相关, 文献中典型选择包括 64×64, 128×128, 224×224, 256×256, 少量研究尝试了 512×512, 768×768 及 1024×1024 等更大尺寸。在将原始二进制文件映射成 r 行 c 列大小的图像像素矩阵后, 实际上已对应到一幅大小为 $r\times c$ 的图像。可采用最近邻插值算法、双线性插值算法等图像缩放方法, 将初始图像规格统一到 $k\times k$, 这里 k 通常选择 64,128,224, 256,512,768,1024 之一。

Zhiguo 等人在将恶意软件映射到图像时, 首先删除像“??”和“00”这样的无意义符号, 其余值则装入到像素矩阵, 图像的纵横比为数值元素数除以图像通道数后取方根, 按 byte-plot 和三通道 byte-plot 方法分别生成灰度图像和 RGB 彩色图像。为解决图像大小不均衡问题, 利用双三次插值算法重构所生成的恶意软件图像[37]。MFF-HDBA 采用双线性插值算法进行恶意软件图像规范化[39]。

3.1.2 像素生成

对于问题二, 我们将其归结为 x-plot 方法, 即定义一组映射函数, 利用原始可执行文件中特定特征 x 及其在文件中的相对位置 i, 生成其在图像中对应的像素点坐标及像素值, 从而将可执行文件转换成一幅一维或二维图像。这里 x 可以是可执行文件的原始字节(称之为 Byte-Plot), 或是可执行文件的头部块中

的信息(如 API 调用, 称之为 API-Plot), 或是汇编程序中的指令码(称之为 Opcode-Plot), 等等。

在生成图像时, 首先要确定图像通道数, 即确定是生成二值图像(Binary image)、灰度图像(Grayscale image)或是彩色图像(Color image), 其次要确定像素值取值区间。通常, 如生成二值图像, 则像素值取 0 或 255(非黑即白); 如生成灰度图像, 可取字节的无符号整数值作为灰度级; 如生成 RGB 三通道彩色图像, 则一般将灰度图像作为其中一个通道, 另两个通道可存储辅助信息, 如文件结构信息、统计信息及熵信息等。

3.1.3 灰度图像生成方法

灰度图像的典型生成方法包括基于原始字节值的 Byte-Plot 方法、基于指令码的 Opcode-Plot 方法以及基于文件结构、块熵、特定对象等信息的其他生成方法。

(1)Byte-Plot 方法。Byte-Plot 灰度图像是目前大多数研究采用的最典型的恶意软件图像映射方法。该方法将文件的原始字节映射到像素。Byte-Plot 方法中, 像素值的取值区间一般为 8 位, 从而可以方便地将可执行程序的每个字节值视作无符号整数直接转换为像素值。Conti 等人最早将 Byte-Plot 方法用于将二进制文件转换为图像, 以在视觉上辅助程序逆向分析[60]。Nataraj 等人最早将该方法引入恶意软件分析[59]。Byte-Plot 灰度图像像素坐标的生成方式有逐行方式(Byte-Plot-Row[40])、往返方式(Byte-Plot-Zigzag[61-63])及网格方式(Byte-Plot-Block[64]), 相关研究中认为后两种方式可以保持更多的字节邻接信息。图 3 是原始二进制文件以 Byte-Plot 模式生成灰度图像的示例。映射时, 一般按表 2 的经验值确定初始图像的宽度 c, 其长度 r 则随文件大小可变。

(2)Byte-Plot 变种方法。Byte-Plot 的变种方法很多, 如: 1D 图像法(Byte-Plot-1D)将二进制文件以向量方式直接映射成一维图像或映射成二维图像后再展平为一维图像, 目的是保证信息连续性[50,65-67]。该方法实际上是展平后的 Byte-Plot-Zigzag 方法。分箱法(Byte-Bin-Plot)根据字节无符号整数值的区间去映射到像素值, 目的是加强图像对比度, 如 Betty 等人将文件字节流中每个字节作为一个 ASCII 字符, 将 ASCII 字符值按不同性质分为 8 组并赋以不同颜色, 再定义模糊集进行色调调整[68]。ASM Byte-Plot 灰度图像(ASM Byte-Plot)先将二进制文件反汇编或反编译成汇编程序或源代码, 再将汇编程序或源代码文件视同一个二进制字节流, 采用 Byte-Plot 方法映射为图像。

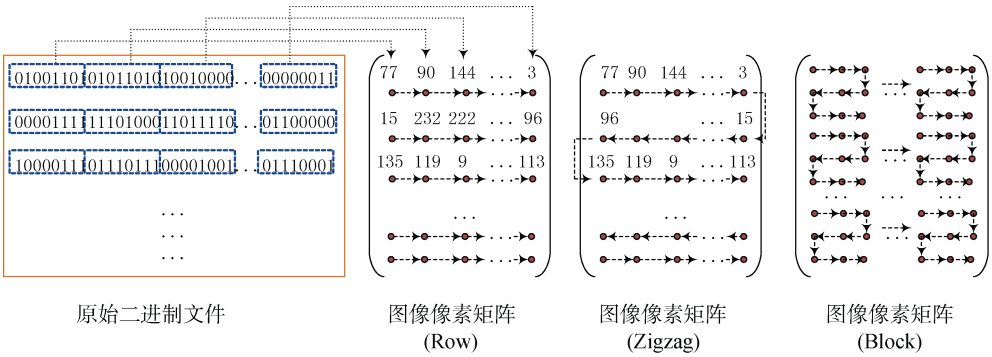

图 3 原始二进制文件到灰度图像像素的映射

Figure 3 Modes for mapping raw bytes into image pixels

(3)Opcode-Plot 灰度图像。Opcode-Plot 方法利用应用程序中的指令码去生成图像, 为此首先用反汇编或反编译工具, 将原始的二进制可执行程序反汇编或反编译成汇编程序或源代码, 抽取出指令码用于生成灰度图像。在具体映射模式上, 又包括以下方式: Opcode-Embedding-Plot 方法: 选出数据集中 Top k 个指令码, 计算其 l 维的嵌入表示, 再为每个程序生成一幅 $k\times l$ 大小的灰度图像。Opcode-Sequence-Plot 方法: 利用 n-gram 指令序列相关信息生成灰度图像。如利用指令码序列的概率和信息增益信息的乘积去生成像素值[69-71]。Opcode-SimHash-Plot 方法: 计算指令码的 SimHash 值, 对 SimHash 的每一个 Bit 位, 视其为 0 或 1 而映射到像素值 0 或 255, 形成一幅二值黑白图像[72]。KyoungSoo 等人[73]利用指令码序列的 SimHash 值去计算图像像素点坐标。

(4)其他方法。生成灰度图像的其他方法包括: Entropy-Plot 方法将二进制文件的字节序列分成固定大小的块, 或者直接将 PE 文件的段作为块, 计算每块的熵值并缩放到 0～255 范围内, 用来生成对应位置的像素值, 所生成的图像又称为熵图(Entropy Image)[51, 74]。Section-Size-Plot 方法对于 PE 格式二进制文件中的每个字节, 利用所在的 PE 段相对于文件长度的比值, 在计算后缩放到 0～255 范围内, 用于生成对应位置的像素值[75]。Byte-Embedding-Plot 采用

Word2Vec 计算二进制文件中每个字节的 256 维向量, 将该文件转换成一个 256×256 维的矩阵, 用于映射成 256×256 的灰度图像[70]。Permission-Plot 方法对于 Android 应用程序, 将其对 144 种权限的请求可视化为一幅大小为 12×12 的图像[76]。APIs-Plot 方法选择 Top k 个 n-gram($n\geqslant 1$) API 对, 以一对 n-gram API 对为图像坐标点, 以其出现的次数作为初始图像的像素值[77]。Markov-Plot 以 256 个字节值作为图像坐标点, 以字节对间的转移概率为初始图像的像素值, 生成的图像又称为 Markov 图像(Markov Image)[78-79]。SFC-Plot 方法提出一种基于空间填充曲线(SFC: space-filling curves)的图像映射框架, 考虑了 Hilbert、Z-order 和 Gray-code 三种空间填充曲线遍历模式, 以此将恶意软件二进制文件映射到 SFC 图像。SFC-Plot 其实可看成 Byte-Plot 的一个变种方法, 亦是将恶意软件文件的二进制代码视为一维线性序列, 再将每个字节比一维空间映射到二维空间, 作者认为 SFC 映射方法可以将二进制文件中的紧密位置点在映射到 SFC 图像中亦趋于紧密位置[80]。

3.1.4 彩色图像生成方法

一些研究人员认为, 灰度图像不能完全包含转换前的恶意软件二进制代码信息, 可以利用彩色图像的多个通道, 在灰度图像基础上, 携带更多的信息。以 RGB 三通道图像为例, 将恶意软件转换成彩色图像的方法包括 Triple-Plot、3-gram Byte-Plot、Colormap-Plot 方法等。

(1) Triple-Plot 方法。该方法利用 RGB 彩色图像的三个通道, 每个通道存储一幅灰度图像, 如图 4 所示。一般用其中一个通道存储 Byte-Plot 灰度图像, 另两个通道存储辅助信息, 如利用 Entropy-Plot 方法得到的熵图、利用 Byte-Embedding-Plot 得到的灰度图像、利用 Section-Size-Plot 方法得到的灰度图像, 以及其他利用文件结构信息或统计信息得到的灰度图像等[4, 81]。Conti 等人[82]将恶意软件二进制文件表示为 3 通道图像, 即将 Markov-Plot、Entrpy-Plot 和 Byte-Plot 灰度图三个通道合成为彩色图像。

(2) 3-gram Byte-Plot 方法。该方法用二进制文件中连续 3 个字节值去生成 RGB 图像的 R 通道、G 通道和 B 通道对应像素的颜色值[75, 83]。方法如图 5 所示。

(3) Colormap-Plot 方法。该方法首先构造一个 $k\times k$ 的 2D 色图矩阵, 其每个元素对应于一个 RGB 值。在实现便利性方面, Matplotlib 库中包含多类自带的 colormaps, 如图 6 所示的感知一致的配色方案(https://matplotlib.org/stable/users/explain/colors/colormaps.html#colormaps)。亦可以基于 Matplotlib 的 colormap 去创建自己的 colormap。Huy 等人对于二进制文件中每个字节, 其前后 4 位分别作为 y 坐标和 x 坐标, 用于查找 16×16 的色图中的 RGB 值并作为待生成的彩色图像的对应像素值[75]。Kesav 等人[61]将转移概率标准化映射到色图, 用其生成像素的颜色值。Danish 等人[84]在将二进制文件映射成 2D 矩阵后, 再基于色图生成彩色图像。MC-ISA 首先将可执行文件

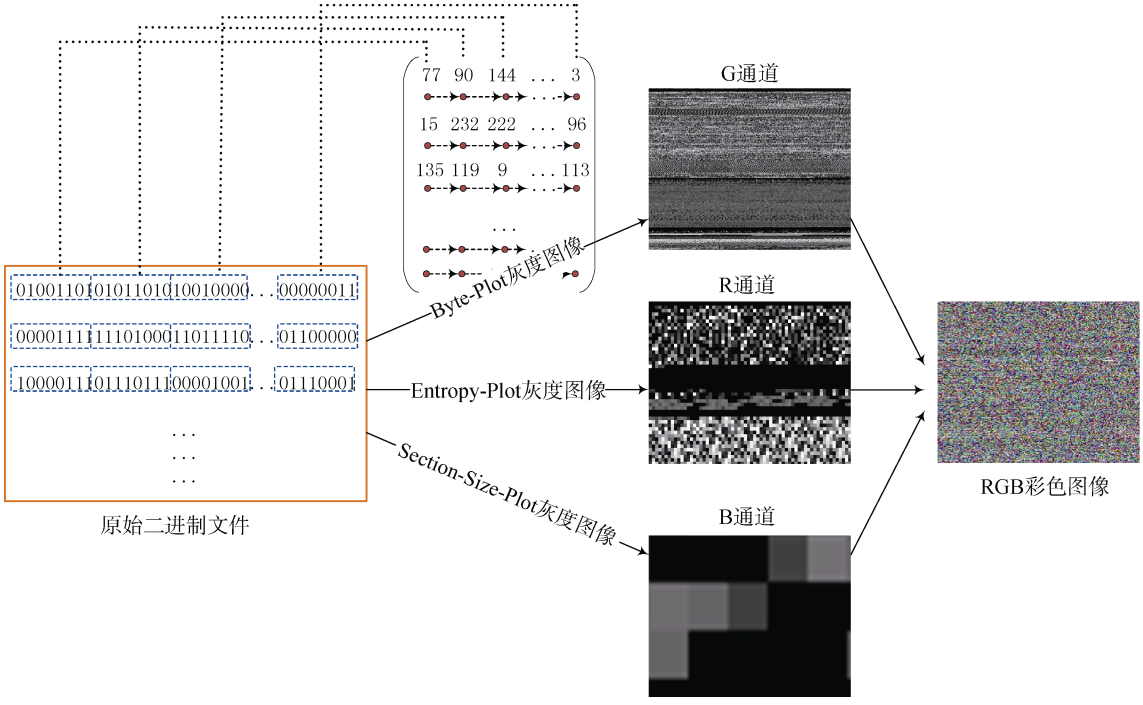

图 4 Triple-Plot 方法

Figure 4 Triple-Plot color image

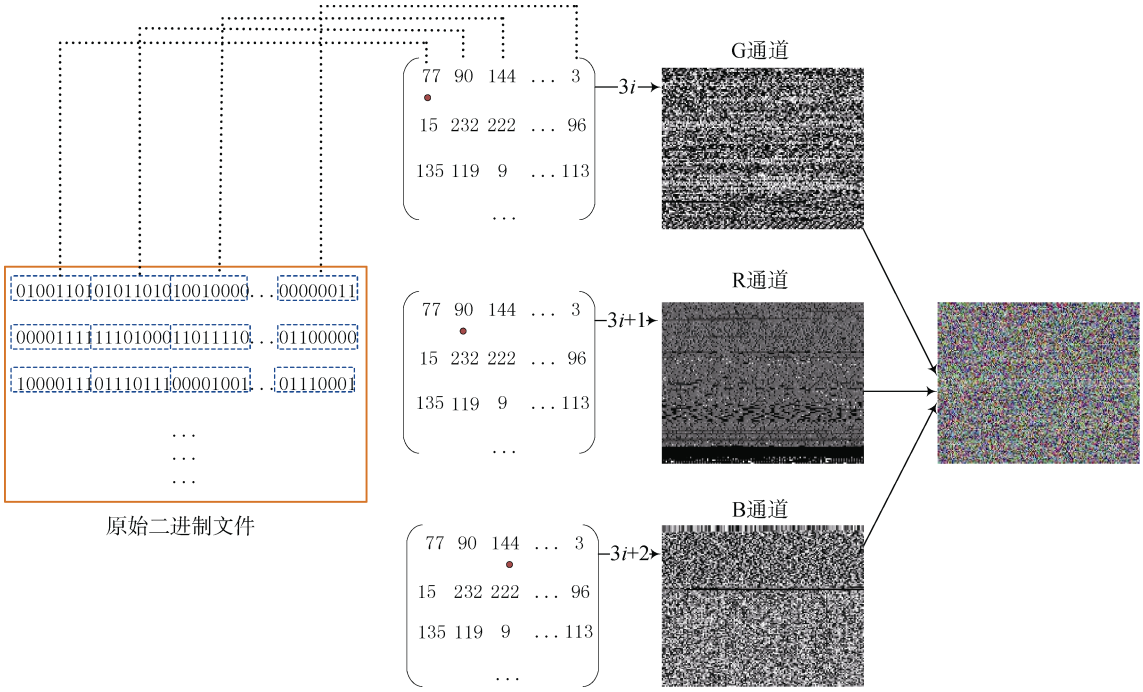

图 5　3-gram Byte-Plot 方法

Figure 5　3-gram Byte-Plot color image

图 6　matplotlib 中一种感知一致的配色方案

Figure 6　Perceptually uniform sequential colormaps in matplotlib

反汇编为汇编程序, 再将汇编文件、二进制文件及其字节序列的 Markov 转移矩阵各可视化为一个图像通道, 用颜色图进行属性增强后生成 RGB 三通道彩色图像[85]。

(4) 其他方法。包括 CoLab-Plot 方法、Datatype-Plot 方法和 Token-gjb2-Plot 方法等。

CoLab-Plot 方法。PE 文件头中段的数量、顺序和大小是 PE 文件的重要属性特征, 如图 7 所示。CoLab-Plot 方法在 Byte-Plot 灰度图像基础上, 利用加色标记盒 CoLab(Colored Label boxes)在生成的图像中标记出 PE 文件的段, 基于段的名字赋以不同的预设颜色, 标记线的粗细则与段的长度相关, 从而在转换后的恶意图像中进一步强调段的分布信息, 所生成的图像称为“CoLab 图像”[86]。

Datatype-Plot 方法。Stephen 等人[80]在颜色编码方案上, 采用数据类型映射, 即字节值 0x00 用黑色表示, ASCII 文本字节(0x01-0x7E)用蓝色表示, 高值字节(0x7F-0xFE)用红色表示, 0xFF 用白色表示, 以此生成彩色 SFC 图像。

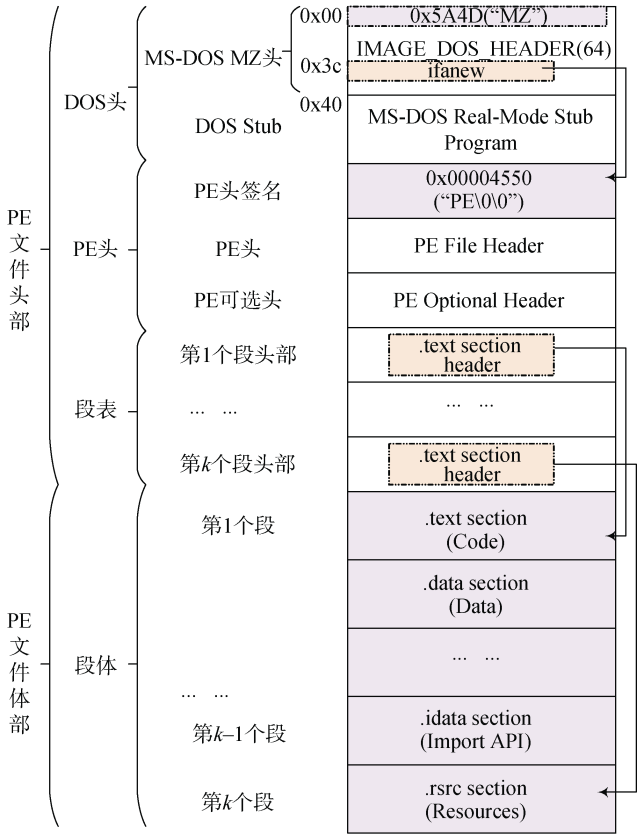

图 7　Windows PE 文件格式

Figure 7　Format of Windows PE file

Token-gjb2-Plot 方法。利用代码段或数据段中的某些特殊字节序列如串、动态链接库或系统调用名等去转换生成初始图像。Yena 等人[87]利用反编译工

具抽取 Android APK 文件中的源代码, 用 TF-IDF 算法计算代码中的词的重要性并用于生成图像, 对每个代码文件中的重要的词组, 利用 SimHash 算法生成初始图像的坐标值, 利用 gjb2 算法生成其红、绿、蓝颜色值。KyoungSoo 等人[73]则对于指令码(类似于词)序列, 用 SimHash 算法生成初始图像的坐标值, 利用 gjb2 算法生成其红、绿、蓝颜色值。

相关研究中典型的图像预表征方法如表 3 所示。总体而言, Byte-Plot 系列方法直接处理可执行文件, 不需专业知识而直接映射成图像, 适于实时处理[88-104]。Opcode-Plot 系列方法首先需要将可执行文件反汇编或反编译为源代码再映射为图像, 不适于实时处理, 且源代码格式会受到反汇编或反编译软件内在逻辑及优化模式的影响, 单一性能反而不如 Byte-Plot 方法。除 3-gram Byte-Plot 彩色图像外, 其他彩色图像一般使用 3 个通道存储三类信息, 效果会优于使用单一通道的灰色图像。Markov-Plot、Permission-Plot 等方法因生成图像携带的信息有限, 一般作为彩色图像的一个通道使用。

表 3 表征方法

Table 3 Representation methods

表征方法	适用范围	文献
Byte-Plot 灰度图像	二进制文件	[59, 63, 70, 88-104]
Byte-Plot-Zigzag 灰度图像	二进制文件	[61-63]
Byte-Plot-Block 灰度图像	二进制文件	[64]
Byte-Plot-1D 灰度图像	二进制文件	[50,65-67]
Markov-Plot 灰度图像	二进制文件/汇编文件	[51,61, 78]
Entropy-Plot 灰度图像	二进制文件	[51, 74]
Opcode sequence-Plot 灰度图像	汇编文件	[69, 71,105]
Opcode SimHash-Plot 灰度图像	汇编文件	[72]
APIs-Plot 灰度图像	汇编文件	[77]
Permission-Plot 灰度图像	二进制文件	[76]
Triple-Plot 彩色图像	二进制文件	[70]
Colormap-Plot 彩色图像	二进制文件	[51, 75, 84]
3-gram Byte-Plot 彩色图像	二进制文件	[75, 83]
Triple-Plot 彩色图像	二进制文件	[4, 75]
SFC-Plot+ Datatype-Plot 彩色图像	二进制文件	[80]
CoLab-Plot 彩色图像	二进制文件	[86]
Token-Plot 彩色图像	二进制文件	[73, 87]

3.2 恶意软件图像特征抽取和选择

面向恶意软件图像分类问题, 其特征抽取主要有两类方法: 一是深度自动提取方法抽取的隐特征(Latent feature), 一般是直接将深度学习分类模型倒数第二层的输出作为特征。二是提取图像专有全局特征或局部特征, 包括纹理特征、颜色特征(颜色矩)、形状特征及统计特征等, 常用的特征提取算子包括: GIST 描述子、Gabor 变换、灰度级共现矩阵(second-order Gray-Level Co-occurrence Matrix, GLCMs)、LBP(Local Binary Patterns)、SURF (Speeded Up Robust Feature)、HOG(Histogram of Oriented Gradients)、SIFT (Scale Invariant Feature Transform)及 D-SIFT (Dense-SIFT)等。

例如: Nataraj 等人[59]用 GIST 描述子计算恶意软件图像的纹理特征并用于分类。Stephen 等人[80]使用 LBP 算子、HOG 算子和 Gabor 过滤器抽取恶意软件图像的特征并用于分类; Vinita 等人[91]提出利用恶意软件灰度图像的 35 个一阶和二阶纹理统计特征进行变种分类, 其中 19 个一阶纹理统计特征包括: 均值、中值、标准差、偏度、峰度、最小值、10 分位、最大值、90 分位、四分位距、平均绝对离差、一阶熵、一阶能量、变异系数、范围、四分位离散系数、绝对中位差、均方根; 二阶灰度共现矩阵的包括: 对比度、角二阶矩、归一化逆差矩、联合平均、离差平方和/联合方差、联合熵、联合最大化、相关度, 取这 8 个特征在 4 个方向上的标准差和均值共 16 个二阶特征; 因此每个恶意软件得到 35 个分类特征, 在 Malimg 数据集上得到 98.58%的分类准确率。Yashu 等人[106]在灰度图像基础上, 进一步融合 GIST 全局特征和 LBP 或 dense SIFT 局部特征, 提高了针对混淆的鲁棒性。

Chenbin 等人[107]利用 GIST、Tamura 等 4 种图像特征提取方法, 提取恶意软件灰度图像的粗糙度(Coarseness)、对比度(Contrast)、方向度(Directionality)、线性度(Linelikeness)、规则度(Regularity)和粗细度(Roughness)等 6 个纹理特征并用于分类, 在 J48、朴素贝叶斯、*k*-最近邻、随机森林和 CNN 共 5 种分类器中, 随机森林取得 93%的最优识别率。Jiang 等人[108]采用 3-gram-Byte-Plot 彩色图像方法, 将二进制文件转换成多通道彩色图像, 采用 AlexNet 提取纹理特征并运用局部响应归一化(LRN)技术, 在 Malimg 数据集上的分类实验获得 97.9%的平均准确率。

iMDA 在 CNN 架构中引入边缘探测和平滑、多路径空洞卷积及通道挤升(channel squeezing and boosting)等模块用于特征学习, 其中边缘探测和平滑方法用于学习恶意软件家族内的局部结构变化; 多路径空洞卷积操作用于识别恶意软件模式的全局结构; 通道挤升用于辅助调节复杂性及得到各种特征图。iMDA 在基准 IoT 数据集上的实验准确率达 97.93%, 较 AlexNet, VGG16/19, Inception V3, ResNet18/50, Shufflenet, DenseNet201, Xception, GoogleNet

等其他 CNN 架构取得更高的检测性能[38]。Sibel 等人[109]利用 CNN 和 LSTM 处理勒索软件的动态特征, 实验表明: 深度学习模型如果使用 API 调用序列作为输入, 虽然真阳率可达 100%, 但假阳率亦很高; 如果使用动态链接库 DLLs 及枚举的目录作为输入, 则可以获得更高的准确率。

相关研究中典型的图像特征抽取和选择方法如表 4 所示。

表 4　文献方法汇总

Table 4　Summary of literature methods

文献	可视化方法	分类特征	分类器	数据集	最优准确率
[4]	[C]: Triple-Plot	GLCM 纹理+颜色矩+可读串 Hash	RF, kNN, SVM	Private [7087/15]	kNN: 97.47%,
[40]	[G]: Byte-Plot	GIST	欧氏距离 k-NN	Malimg	Malimg: 98.00%
[63]	[G]:Byte-Plot (zigzag) [C]:Markov+Colormap-Plot	Gabor, 小波和强度特征	SVM, Random Forest	Private [5000/9]	Markov+Colormap-Plot: 检测 95.00%, 鉴别 81.00%
[64]	[G]: Byte-Plot(zigzag); Markov-plot	LBP + 直方图	k-NN	D-I: [1000/?] + [1000/0]; D-II: [1600/?] + [1600/0]; D-III: [1000/?] + [1000/0]	Markov -Plot-D-I, D-II, D-III: 85.39%, 95.23%, 91.39% Byte-Plot-D-I, D-II, D-III: 85.54%, 93.84%, 86.09%
[65]	[G]: Byte-Plot(block)	LBP	Tensorflow CNN, SVM, k-NN	Malimg + BIG 2015	93.17%
[66]	[G]: Byte-Plot-1D	隐特征	CNN, CNN-UniLSTM, CNN-BiLSTM	BIG 2015	CNN-BiLSTM: 98.20%
[51]	[G]: Byte-Plot [G]: Byte-Plot-1D	隐特征	MLP, CNN, Res-Net152, VGG-19	Malicia + Samuel Kim [26413/20]	VGG-19: 92.16%
[67]	[G]: Byte-Plot-1D	隐特征	两层 CNN+LSTM pipeline	D-I: Malimg; D-II: Private [10350/10]	D-I: 96.30%; D-II: 98.80%
[68]	[G]: Byte-Plot-1D	预训练 CNN 抽取隐特征	CNN, CNN biLSTM, and CNN BiGRU	BIG 2015; Malimg; D-III: Private [52000/?]+[23000/0]	BIG 2015: N/A; Malimg: 96.30%; D-III: 98.60%
[70]	[G]:Opcode-Sequence-Plot	隐特征	GoogleNet Inception V4; ResNet	BIG 2015+ Private [3000/0]	ResNet 152: 88.36%
[71]	[C]: Triple-Plot	隐特征	LeNet5	BIG 2015	98.76%
[72]	[G]:Opcode-Sequence-Plot	膨胀+腐蚀后直方图	3 层 CNN	Private [9168/10] + [8640/0]	96.70%
[73]	[G]:Opcode-SimHash -Plot	隐特征	CNN	BIG 2015	98.86%
[74]	[G]: Token-gjb2-Plot	N/A	选择性区域匹配	Private [8169/3]+[2505/0]	同家族至少 0.95 不同家族低于 0.325
[52]	[G]: Colormap-Plot [C]: Colormap-Plot [G]: Markov-Plot	Gabor filter	(检测)基于 CNN, VGG3, Resnet50 架构的 12 种模型	D-I: BIG 2015 + [12971,0]; D-II: MalImg + [12971,0]	D-I: 99.20%; D-II: 99.95%
[52]	[G]: Entropy-Plot	块熵, 像素直方图, 块熵均值和标准差	(分类)VGG3; 自定义 7 层 DNN	BIG 2015; Malimg	VGG3-BIG 2015: 98.25%; VGG3-Malimg: 98.46%
[75]	[G]: Entropy-Plot	CNN 抽取隐特征	SVM	Malimg; BIG 2015	Malimg: 99.70% BIG 2015: 100%
[76]	[G]: Byte-Plot [C]: Triple-Plot; 3-gram Byte-Plot; Colormap-Plot	隐特征	kNN, SVM, MLP, RF, XGBoost, AC-GAN, Resnet152 及集成分类器	MalExe [26412/20]	Colormap-Plot+Resnet152: 91.39%
[77]	[G]: Permission-Plot	隐特征	LeNet, GoogleNet	NCSU: [1200/?] Drebrin: [5560/?]; D-III: [700/0]	LeNet: 93.00% GoogleNet: 92.00%
[78]	[G]: APIs-Plot	自编码器抽取隐特征	自编码器, J48, MLP 和朴素贝叶斯	Private [22000/0] + [26000/?]	95.00%
[79]	[G]: Markov-Plot	隐特征	CNN	BIG 2015; Drebin	BIG 2015: 99.264% Drebin: 97.364%
[84]	[C]: 3-gram Byte-Plot	隐特征	AlexNet, VGG, GoogleNet, Incep-tion-v3	Private [20000000/?0]	Inception-v3: 98.4225%
[85]	[C]: Colormap-Plot	隐特征	fine-tuned CNN	Malimg; IoT-Android: [14733/?] + [2486/0]	Malimg: 98.82%; IoT dataset: 97.35%

续表

文献	可视化方法	分类特征	分类器	数据集	最优准确率
[87]	[C]: Byte-Plot + CoLab-Plot	隐特征	VGG16, DT, kNN, SVM	BIG 2015	CoLab+VGG16+SVM: 96.59%
[88]	[C]: Token-gjb2-Plot	隐特征	CNN	Private [720/?] + [720/0]	92.67%
[89]	[G]: Byte-Plot	文件大小、熵、对比度、能量、均匀性等	自定义两层 ANN	Malimg	99.13%
[90]	[G]: Byte-Plot	隐特征	CNN, bidirectional LSTM	OBF [18800/47], BIG 2015, Malimg	BiLSTM-OBF: 93.40%; BiLSTM-BIG 2015: 93.80%; BiLSTM-Malimg: 98.50%
[91]	[G]: Byte-Plot	隐特征	ResNeXt50	Malimg; modified Malimg	Malimg: 98.32% modified Malimg: 98.86%
[92]	[G]: Byte-Plot	一阶纹理+ GLCM 二阶纹理统计	集成学习分类器	Malimg; D-II: [2916/19]	Malimg: 98.58% D-II: 98.11%
[93]	[G]: Byte-Plot	隐特征	kNN, ResNet34	D-I: Malimg; D-II: Malicia [9895/51]	ResNet34-Malimg: 94.80%; kNN-Malicia: 99.43%
[94]	[G]: Byte-Plot; ASM Byte-Plot	D-SIFT + GIST	SVM, NB (Naïve Bayes), k-NN	Malimg + Malheur ([3131/24]) + VirusShare ([2630/11]) + BIG 2015 子集 [3237/?]	98.20%
[95]	[G]: Byte-Plot	隐特征	CNN	Malimg; BIG 2015	Malimg-256: 98.48%; BIG 2015-256: 97.49%
[96]	[G]: Byte-Plot	隐特征	CNN	Malimg	Malimg-96: 94.50%
[97]	[G]: Byte-Plot	隐特征	基于 VGG-16 的 CNN	BIG 2015; Malimg	Malimg: 98.52%; BIG 2015: 99.97%
[98]	[G]: Byte-Plot	Gabor 小波+ GIST	基于前向传播 ANN	Mahenhur [3131/24]	96.35%
[99]	[G]: Byte-Plot	熵图	直方图相似度	Private [1000/50]	97.90%
[100]	[G]: Byte-Plot	隐特征	VGGNet CNN, BaseNet CNN	BIG 2015+ [20650/0]	VGGNet CNN: 98.14%; BaseNet CNN: 96.82%
[101]	[G]: Byte-Plot	隐特征	基于预训练 DNN 的迁移学习	Private [197604/0] + [584606/?]	99.07%
[102]	[G]: Byte-Plot	颜色直方图, Hu 矩, Haralick, SIFT, SURF, ORB, KAZE	CNN (1D, 2D, 类 VGG16)	Private [4850/?] + [4850/0]	SURF: 98.96%; SIFT: 97.62%; KAZE: 97.25%; ORB: 93.53%; 全局特征: 96.32%
[103]	[G]: Byte-Plot	图像纹理 + API 调用	深度信念网络	D-I: Drebin 子集[1550/?], [2620/?], [5825/?], [6965/?] D-II: BIG 2015 子集	D-I 的[5825/?]: 95.70% D-II 的[3626/?]: 97.24%
[104]	[G]: Byte-Plot	CLAHE	CNN	Malimg	96.00%
[105]	[G]: Byte-Plot	隐特征	CNN, SVM	BIG 2015; Ember 2018	97.73%
[106]	[G]:Opcode-Sequence-Plot	隐特征	CNN	BIG 2015	98.00%
[107]	[C]: SFC-Plot+ Data-type-Plot	LBP, HOG, Gabor filter	RF, SVM and KNN	Private [13599/23]	混合 kNN-HOG Z-order: 97.6%
[108]	[G]: Byte-Plot	GIST+LBP/Dense SIFT	RF	Malimg; Antiy [11000/11]	Malimg: 96.60% Antiy: 94.98%
[109]	[G]: Byte-Plot	GIST, Tamura	J48、朴素贝叶斯、kNN、RF 和 CNN	Private [?/?]	RF: 93.00%
[110]	[C]: 3-gram Byte-Plot	隐特征	AlexNet	Malimg	97.90%
[121]	[G]: Byte-Plot	隐特征	Attention + Dense-Net-BC	Malimg	94.64%

(注: “可视化方法”列中, [C]和[G]分别表示彩色图像和灰度图像。“数据集”列中, 除 Malimg、BIG 2015、Ember 2018 等通用数据集外, 其他数据集采用 DSname[*m*/*k*]表示, 其中 DSname 为数据集名, Private 表示私有数据集; 当 $k>0$ 时表示包含 k 个恶意软件家族共 m 个恶意软件样本; 当 $k=0$ 时表示包含 m 个良性程序样本。)

3.3 恶意软件图像分类

相关文献中恶意软件图像分类模型主要包含机器学习模型和深度学习模型, 其中深度模型又包含定制网络架构和预训练网络。

如果特征提取阶段提取的是图像专有全局特征或局部特征, 如纹理特征、颜色特征(颜色矩)、形状特征及统计特征等, 则常用机器学习模型亦可以取得较好的分类效果。CSM 模型(Conglomerate Stratum

Model)包含 TST(Triad Seeped Technique)、QLM (Quatrain Layer Method)和 APS(Acclimatized Patronage Scheme)三个部件, 分别用于恶意软件图像正则化、图像特征抽取和分类, 在 Malimg 数据集上的实验取得 99.41%的分类准确率[110]。Farhan 等人首先利用来自 Transformer(BERT)的双向编码器表示预训练模型, 基于网络流的嵌入表示抽取训练纹理特征, 同时采用恶意软件-图像转换算法将网络字节流转换成视觉表示, 利用 FAST(Features from Accelerated Segment Test)抽取方法和 BRIEF(Binary Robust Independent Elementary Featrue)描述子抽取及标记视觉特征。将训练纹理特征和视觉特征组合后利用 SMOTE (Synthetic Minority Over-Sampling)方法进行数据平衡, 用 CNN 模型挖掘深度特征后, 采用 Gaussian Naïve Bayes、支持微量机、决策树、逻辑回归和随机森林构成的集成模型进行分类和检测, 在 CICMalDroid 2020 和 CIC-InvesAndMal2019 两个安卓恶意软件数据集上的实验结果表明了该方法的有效性, 精度、召回率、F1-Score 和准确率分别为 99%、99%、99%和 99.16%, 优于对比的 RNN、LSTM、DNN 和 GRU 方法, 其中 DNN 次优, 而最差的 RNN 方法分别只有 85%、85%、87%和 85.34%[111]。MLP-Mixer-Autoencoder 是一个组合了自定义 MLP-mixer 和自编码器的集成架构, 利用自编码器的编码器-解码器架构去精选 MLP-mixer 所抽取的特征[112]。Katrina 等人利用 PIL(Python Image Library)将恶意软件可执行码分别转换成黑白图像、灰度图像、RGB 图像、CMYK 图像和 YCbCr 图像。利用从 Malicia 项目获取的恶意软件家族数据, 在每个家族中掺入另一家族的少量图像, 在其上比较 CNN 分类器和基于 GIST 描述子特征的多种分类算法(*k*-最近邻、随机森林、SVM 和多层感知器)的检测性能, 发现就恶意软件混淆现象, GIST 描述子较 CNN 更为鲁棒[113]。SFTA-GDA 方法结合 SFTA(Segmentation-based Fractal Texture Analysis)和 GDA(Gaussian Discriminant Analysis)用于恶意软件检测, 首先将 MaleVis 数据集的彩色图像转换为灰度图像, 从中抽取 SFTA 和 Gabor 特征, 用 GDA 和朴素贝叶斯进行分类, 达到 98%的准确率[114]。MDABP 在宿主机操作系统中截持虚拟机生成的系统调用作为动态特征, 使用 *k*-NN 分类模型检测跨体系架构的 IoT 恶意软件, 在实验数据集上实现 97.18%的平均准确率, 优于基于网络流的最优跨架构检测方法取得的 94.5%的准确率[115]。

对于深度抽取的恶意软件图像特征, 因 CNN 及其变种以其在图像处理领域的成功, 亦成为恶意软件图像分类的主要模型[116]。MCTVD 设计了一个轻量级卷积神经网络对恶意软件图像进行分类, 在 BIG 2015 恶意软件数据集上分类准确率可达 99.44%, 效果优于 AlexNet、VGG16、VGG19 等通用 CNN[81]。DSBEL 恶意软件检测框架包含基于空洞卷积 CNN 的深度特征提取和基于多分类器的集成学习两部分, 其中 CNN 采用 SB-BR-STM(Squeezed-Boosted Boundary-Region Split-Transform-Merge)架构, 通过在 STM 块中引入通道 Squeezed-Boosted 方法有助于捕获恶意软件图像中更小的纹理和对比度变化; 集成学习使用了 SVM、MLP 和 AddaBooSTM1 三个分类。DSBEL 框架对总共 3959 个 IoT 物联网恶意软件图像的数据集(其中恶意软件图像 1473 个), 实现 98.50%的检测准确率, 优于 SqueezeNet、ShuffleNet、Inception V3、VGG-16、ResNet-50、GoogleNet、DenseNet201 等对比模型[117]。Park 等人[118]提出一个用于恶意软件检测的视觉 Transformer, 并用多个块的附加编码得到用局部特征的位置信息及其间关系去增强该 Transformer, 从而[CLS]可以用于总结所有信息。在 BIG 2015 数据集上的实验表明该模型的准确率优于对比模型。Yimeng 等人[119]对 Malimg 灰度图像, 结合 DenseNet-BC 网络和注意力机制构建恶意软件家族分类模型, 实现 94.64%的分类准确率。Muhammad 等人在 CNN 架构设计上, 结合使用多级特征融合(MFF)和通道注意力机制进行更具区分性和鲁棒性的特征抽取。针对传统超参数优化方法需要人工调整这一不足, 提出一种基于 Bat 算法的超参数优化算法(HDBA)。在 Malimg 数据集和 DataCon 数据集上的实验结果表明, 所提出的模型对于 256×256 大小的图像准确率最高, 可达到 99.36%, 同时该模型可以更有效地鉴别恶意软件变种, 较对比方法 AlexNet8(94.12%)、VGG16 (97.11%)、ResNet50 (97.54%)、Inception V3(98.71%)更高[39]。Yanli 等人[120]在深度残差网络中引入混合注意力机制构建分类模型。Xuelei 等人[121]提出一种融合 Efficient-Net 和 1D-CNN 的恶意软件家族分类方法, 将恶意软件反汇编为汇编语言文件后再可视化为灰度图像及表示成 1 维向量, 分别输入两个网络, 两维卷积网络用于抽取恶意软件的纹理特征, 1 维卷积用于抽取局部邻接信息, 两个网络在反向传播时同时调整, 在所爬取的 VirusShare 数据集(38 类)和 Malimg 数据集上的实验表明, 最优准确率分别为 98.14%和 98.55%。

预训练模型拥有庞大网络连接和巨量模型参数, 隐式编码了丰富的领域知识, 通过微调可以迁移到其他领域的下游任务, 可视化恶意软件分类任务也

可借鉴利用。Pratikkumar 等人将预训练的 ResNet152 和 VGG-19 模型迁移到恶意软件图像分类任务, 实验表明两个预训练模型的分类性能均优于基于图像的 MLP 和 CNN 模型以及基于指令码的 CNN、RNN、LSTM 和 GRU 模型。Mohamad 等人[123]提出一种基于蝶形结构的视觉 Transformer 模型 B_ViT, 用于基于恶意软件图像分类。B_ViT 在实现上采用了一个在 ImageNet 数据集上预训练的 ViT 进行迁移学习, 在 Malimg、BIG 2015 数据集上的实验表明, B_ViT 的分类准确率分别达到 99.49%和 99.99%, 优于对比的 IEViT(Input Enhanced ViT)和 ViT 模型。Osho 等人[123]利用预训练的 Inception-ResNet-V2 CNN 模型进行恶意软件灰度图像分类, 在 Malimg 数据集及自建数据集上均取得 99.2%的分类准确率。EfficientNetB1 在 BIG 2015 数据集生成的 Byte-Plot 灰度图像上, 可取得 99%的分类准确率, 较多种基于 CNN 的预训练方法性能相当但网络参数更少[124]。MC-ISA 在引入图像大小自适应、颜色增强和多通道增强后, Inception-ResNet-V2 的检测准确率最高(99.36%), 优于预训练的 ResNet50、Inception V3、VGG16[85]。

集成模型也体现了一定优势。Farhan 等人[111]采用 Gaussian 相素贝叶斯、支持向量机、决策树、逻辑回归和随机森林构成的集成模型取得较好分类效果。Pratyush 等人[125]使用 CNN 自动抽取恶意软件灰度图像的特征, 送入由自编码器、GRU、MLP 构建的 Stacked Ensemble 集成模型 SE-AGM 训练, 对 Malimg 数据集, 当训练集与测试集数据比例为 8:2 时, SE-AGM 实现 99.43%的平均测试准确率, 略优于 7:3 时的 99.30%。

另外, 针对恶意软件家族之间数据失衡、部分家族样本较少的问题, 也有研究探索应用少样本学习方法。Conti 等人[82]利用两种少样本学习架构 CSNN (Convolutional Siamese Neural Network)和 Shallow-lFS(Shallow Few-Shot)分类方法处理新型恶意软件家族。在 Malimg、BIG 2015 和 MalwareBazaa 三个数据集上进行实验, CSNN 分别取得 96.21%, 94.99%和 93.42%的分类准确率, Shallow-lFS 在 10-shot 时分别取得 98.26%, 88.68%和 97.65%的分类准确率, 较对比方法取得更高。

总体来看, 对于图像专用特征, 典型的机器学习方法 *k*-最近邻、SVM 等即可得到较好分类效果。如果采用深度模型直接提取分类特征, 则多采用 CNN 架构网络, 定制设计和预训练模型微调均可获得较好的分类效果, 但如果不加微调, 预训练模型的分类效果会受影响[50]。

3.4 恶意软件图像分类结果评估

可以从分类模型和分类效果两个方面评估分类器, 其中模型评估主要是基于训练数据, 评估分类模型预测值与实际值拟合的好坏程度, 并可进一步通过超参数优化开展模型泛化能力和鲁棒性评估, 相应的指标包含平均绝对误差(MAE: Mean Absolute Error)、相对绝对误差(RAE: Relative Absolute Error)、均方根误差(RMSE: Root-Mean-Square Error)等的实际评估两个方图像质量; 分类效果评估主要是基于测试数据, 评估正、负样本中被分类模型正确预测相应类别比率, 相应的指标包含准确率(Accuracy)和总体准确率、精确率(Precision)、召回率(Recall)、F_1-Measure 综合指标等, 受数据集规模及样本类别分布限制, 往往通过三次分割(three-way split)方式将数据集划分为训练集、测试集和验证集, 并选择使用 leave-one-out 交叉验证或 k 折交叉验证以得到相对稳定的效果指标。在效果评估时, 亦可以依据正负样本比例变化, 引入 ROC 曲线和 AUC 面积对分类效果进行动态评估。

针对基于图像的恶意软件分类器, 文献所采用大多数模型均是成熟的 ML 或 DL 模型, 这些模型的功效基本上都在图像处理任务中得到了验证, 因此影响模型性能的主要就是图像的质量, 故基于图像的恶意软件分类器的模型评估任务可以对应转换为恶意软件生成图像的质量评估任务。MIGAN 是一个基于 AC-GAN 的恶意软件图像数据增加模型, 在 Malimg 上的实验表明 MIGAN 能够得到增广的标注数据集, 生成的图像样本的质量分布采用结构相似性指数(Structural Similarity Index, SSIM)和 FID (Ffechet Inception Distance)进行度量[126]。FENCEBOX 框架提供了三大类 15 种图像数据增强方法: 畸变、压缩和噪声注入, 再采用 L2 范数、SSIM 和峰值信噪比(Peak Signal-to-Noise Ration, PSNR)去度量这些方法所生成的数据的质量[127]。

针于基于图像的恶意软件分类效果评估, 因准确率代表了被分类器正确预测的样本数占总样本数的比例, 几乎被所有文献作为必备的性能指标之一。相关文献取得的最优准确率如表 4 所示。

4 优势、问题与挑战

基于图像可视化的恶意软件分类技术将恶意软件映射成图像, 从而将恶意软件分类任务转换成图像处理任务, 此方法具有三个明显的优势:

一是有利于缓解对专家知识的强依赖。传统的恶意软件分类技术高度依赖领域专家去分析提取恶

意模式, 必须结合文件格式、指令系统、恶意行为甚至编译系统的深度专业知识。在当前恶意软件数量爆发式增长的情况下, 人工辅助特征工程的难度不断提升, 而恶意软件图像则以对人类更为直观可感知的方式, 提供了探索恶意模式的新途径。

二是更适用于恶意软件变种的检测。恶意软件开发者为了以最小代价逃避恶意软件检测, 往往进行微小的代码变化, 特别是对于自动或人工智能生成的对抗样本, 其代码较原始代码有着微小扰动, 反映在生成图像上, 也表现为图像的细微变化, 易被基于可视化的恶意软件分类模型识别。

三是可以借鉴图像处理领域的系列研究成果。总体来看, 人类对图像处理技术的研究远较恶意软件分类技术广泛而成熟, 特别是深度图像处理技术取得了瞩目成果, 将其引入恶意软件分类领域, 既提供了新的探索思路, 也有利于提供预训练的模型供迁移学习。

综观近 10 多年来基于可视化恶意软件分类技术的学术研究, 仍聚焦于相适应的特征表示、特征抽取、特征选择和分类方法, 在典型数据集上均取得了较好的检测性能, 但效果难以进行消冗性解释。基于图像可视化的恶意软件分类技术所面临的问题既有各种恶意软件分类方法所面临的共性问题, 亦有个性问题。主要包括:

一是恶意载荷定位困难问题。恶意软件文件本身存在宏观结构与微观结构包容重叠, 原始表示的意义呈现多态性, 如原始字节数据可以是字节码、指令码、操作数或任何其他对象之一部分, 字节间关系模糊, 缺乏人类直觉或自然特征。而且无论恶意可执行文件还是良性可执行文件, 均由长度不等的指令序列组成。恶意软件文件中一般也只有少量代码序列具有恶意行为, 这些恶意行为形式多样, 组合灵活, 对应的指令序列甚至与某些良性文件中的一些指令序列极其相似, 需要极强的领域专家知识方能准确定位, 采用 *x*-plot 转换成图像后亦然。例如, MALIGN 采用核酸和指纹对恶意软件进行编码, 将恶意软件家族分类问题转换成基因序列对齐问题, 该方法有助于鉴别恶意软件家族的恒定块, 但仍无法确定二进制文件中是否存在以及在何处存在足以代表该家族的恒定块及其恶意性[128]。

二是模型适用性问题。与视觉领域相比, 对可视化方法在恶意软件分类或分析中的适用性缺乏系统理论支撑和比较研究。将恶意软件二进制文件映射为图像, 自然期望其中的恶意载荷被映射为图像中一块连续的像素区域, 从而可以通过基于深度学习模型去挖掘出这个代表性的区域。然而事实是恶意软件源代码经编译甚至混淆, 其中的恶意载荷已被稀释到整个代码中。由于自然图像中的目标对象往往呈块状区域, 相似目标对象往往表现为块状区域在可感知程度内的扭曲或信息损失, 且相对整幅图像的熵值变换相对较低, DL 模型能够敏感地鉴别到相似性。然而, 恶意软件一旦转换成图像, 其恶意载荷往往呈现为图像中的线段, 恶意软件家族检测转换成未知位置处若干线段的相似性检测, 模型可使用的判别特征显著减少。即使对于恶意软件家族分类, 也难以确定模型究竟是依赖家族恶意载荷相似性还是基于家族文件结构相似性做出了判定。

三是结果可解释性问题。对于基于机器学习和深度学习的领域分类任务, 如果对象表征的可解释性越好, 则模型分类结果的可解释性也越好, 对表征方法和模型调优的方向也越明确, 分类性能改善效果也越明显。但现有基于图像可视化的恶意软件分类研究文献中, 分类实验结果评估的科学性和可解释性都有待进一步从多方面探究, 例如对于可视化对象和模式对分类效果的影响、基于可视化方法的恶意特征或家族特征发现程度、图像特征工程对分类效果的影响分析、基于深度模型的恶意载荷发现定位、分类模型泛化的影响因素, 等等, 都缺乏系统深入的探索。

四是高质量标注数据稀缺问题。任意一个可执行代码文件, 除非已知恶意载荷字节确定性特征, 否则很难确定该文件是恶意文件还是良性文件。恶意软件鉴别需要极强专业知识而难以规模化采集标注, 相关数据深度涉及用户通信隐私而难以公开发布, 导致目前业界公开可用数据集零星陈旧。文献中使用最多的数据集包括 Windows 平台 PE 恶意软件的 BIG 2015 数据集、Malimg 数据集和安卓平台 APK 恶意软件的 Drebin 数据集, 为避免被滥用, 这些数据集公开发布时均进行了转换或脱敏处理, 不可避免造成原始特征损失和利用模式受限, 且因发布时间较早, 已难以代表恶意软件的最新技术路线、家族特征及行为模式。MALNET-IMAGE 是一个较新的百万级恶意软件图像集, 包含 47 类 696 个恶意软件家族超过 120 个恶意安卓软件图像, 但无法逆向成原始 APK 文件[129]。利用这些数据集训练得到的分类器难以进行工程化部署和结果解释。

5 未来展望

与其他领域相比, 恶意软件分类既有样本数据稀少带来的空间稀疏离散问题, 也有家族类别差异

带来的数据天然失衡问题; 既有原始表示缺乏直觉或自然特征带来的恶意载荷定位困难问题, 也有原始字节数据可以是字节码、指令码、操作数或任何其他对象之一部分, 恶意软件本身存在宏观结构与微观结构包容重叠带来的意义多态性问题; 既有普遍使用打包、加密等技术带来的非对抗混淆问题, 也有被攻击者故意破坏格式规范、操纵代码或深度混淆带来的主动对抗问题。未来恶意软件分类方法应充分考虑恶意软件的这种结构特征性和内在对抗性。可能关注的研究方向包括:

一是基于认知的 DL 模型。现有 DL 模型内在地是贴近人类感知而设计, 更适合于自然语言、图像、音视频等这类大众泛知对象的处理任务, 利用足够大的模型和足够多的训练数据, 这在这些任务上已能够达到不弱于人类的处理性能。但对于可执行软件这类善恶难以判定, 恶意载荷位置难以确定, 恶意行为难以鉴别, 需要深度专业知识才能分析的目标对象, 难以自动提取可供模型感知的关键特征。且因安全和隐私原因, 无法获得足够多的训练数据, 导致训练得到的模型存在经验与现实之间无法度量的偏差。究其原因, 当前基于感知的 DL 模型只能基于已有经验或知识, 无法适应恶意软件的频繁变化和内在对抗。未来研究中可考虑基于认知的 DL 模型, 利用现有知识生成新的知识并用于决策, 有助于实现恶意软件的知识获取、理解、推理和解释。

二是恶意软件领域知识图谱。现有恶意软件检测系统普遍受到代码混淆技术的影响, 变种恶意软件生成与检测技术之间存在不成比例的攻防对抗竞赛。这些混淆既有面向菜鸟级恶意软件开发者的打包、加密等多态类操作, 又有高级恶意软件开发者为恶意逃避而采用的代码隐藏、重写、操纵、填充等变态类操作, 更有为逃逸 DL 型检测系统而专门定制的对抗样本。这种普遍对抗性使得新型恶意软件层出不穷, 而检测上总存在时间滞后性。未来可考虑研究建立恶意软件领域知识图谱, 利用恶意软件开发者、代码特征、家族基因、行为特性、传播方式, 以及通用第三方软件包调用等相关知识, 提高对变种恶意软件和新型恶意软件的检出能力。基于图像的恶意软件表示方法可用于知识图谱构建中恶意软件家族基因总结和抽取。

三是恶意软件样本对抗增强。对于小样本的自然图像数据类别, 为解决不同类别之间的样本数据不均衡问题, 一种方法是采用动态重采样扩增小类别数据, 但缺点是无法增强扩增类别数据的代表性。更好的方法是采用可控程度的旋转、缩放、平移、加噪、调色等通用图像变换等方法去扩增小样本数据类别。但对于恶意软件来说, 图像变换方法会带来显著的信息扭曲或损失。因为恶意软件在问题空间极其稀疏离散, 离散域使得扰动成为一个不可微或非凸的任务, 无法直接使用基于梯度的方法去指导扰动方向及幅度。同时恶意软件功能存在内生约束性, 修改其中某个字节可能导致程序崩溃或恶意功能失效, 在特征空间扰动后的样本很大可能无法映射回问题空间中的自然恶意软件, 因此为保全功能就必须限制可行扰动操作和扰动幅度。相较对抗样本生成是扰动恶意软件去生成一个可逃避检测的疑似良性样本, 恶意数据对抗增强是扰动恶意软件去生成尽可能多的恶意样本, 对于生成模型的架构、参数选择和优化控制都提出新的要求。未来可考虑基于大规模二进制程序函数相似性分析技术以及基于大型语言模型生成技术, 筛选出功能等价代码段并用于增强恶意软件数据。同时, 基于图像的恶意软件表示方法可用于大规模二进制代码段筛选和语义信息挖掘。

四是评估基准与高质量数据集。深度学习本质是基于训练数据去发现数据中隐含的特征和规律, 数据规模和质量是保证模型功效的重要条件。而网络安全数据内嵌用户隐私而极其敏感, 即使脱敏处理也可能遭受记录恢复或画像攻击, 基准数据集的发布面临包括法律限制、标注困难、安全义务和预防措施等方面的挑战, 有限可用的几个公用数据集不是过时就是存在缺陷, 甚至包含大量无用信息或是误导性特征[90]。真实数据的标注既需高级专家知识也特别耗时, 一是研制标注至恶意载核粒度或其恶意图像像素块粒度的精准数据集; 二是可考虑以众包方式, 迭代生成高质量标注数据集, 通过安全接口提供数据共享服务, 对于促进深度学习在网络安全应用中可行部署具有重要实用意义。

6 总结

恶意软件与良性程序的差异仅是其内嵌有恶意载荷代码序列, 该序列对应着恶意行为。可执行程序生成过程中, 其源代码中恶意载荷所蕴含的清晰语义因编译过程而模糊化, 因编译器配置参数设置不同而多样化, 更因广泛使用的加密、多态、变形、混淆等技术而进一步随机化, 这一系列操作使得恶意软件特别是未知新型恶意软件的检测面临巨大挑战。由于可执行程序这种具有字节、指令码、控制流图等不同粒度级上相互嵌套、内容与功能表里相依的语义, 有效检测特征的定位和提取非常困难。基

于深度学习的图像处理技术因其强大的图像隐含特征定位和检测能力, 为恶意软件分类提供了一条新的思路: 即将人类不可感知的二进制码可执行程序映射成图像, 来自同一恶意软件家族的恶意软件因其结构相似性或恶意载核相似性, 映射为图像后较非同一家族的恶意软件或良性程序具有人类可感知的视觉相似性, 从而可以借鉴当前获得巨大成功的深度图像处理技术, 实现恶意软件的分类。

本文对十几年来基于图像可视化的恶意软件分类研究文献进行系统性研究。鉴于基于机器学习或深度学习模型的图像分类任务已得到广泛验证, 因此将研究聚焦于恶意软件的图像生成方法, 本文将其系统性归结为 *x*-plot 方法并对应分析了不同类型图像的生成, 同时从总体上研究了恶意软件通用分类框架及其基于图像可视化方法的框架实例。

基于图像可视化的恶意软件分类方法, 既有通用恶意软件分类方法所面临的共性问题, 亦有因图像可视化而引入的个性问题, 包括: 恶意载荷定位困难问题、模型适用性问题、结果可解释性问题以及高质量标注数据稀缺问题。面对这些问题及挑战, 展望了未来几个可行且重要的研究方向, 包括: 基于认知的 DL 模型、恶意软件领域知识图谱、恶意软件样本对抗增强以及评估基准与高质量数据集。

参考文献

[1] Eric M. Hutchins, Michael J. Cloppert, Rohan M.Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. https://www. lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/ LM-White-Paper-Intel-Driven-Defense.pdf. May 2021.

[2] CNCERT/CC. First half year cybersecurity analysis report in 2021. https://www.cert.org.cn/publish/main/upload/File/first-half%20%20year%20cyberseurity%20analysis%20%20report%202021.pdf. July 2021.
(国家计算机网络应急技术处理协调中心. 2021 年上半年我国互联网网络安全监测数据分析报告. https://www.cert.org.cn/publish/main/upload/File/first-half%20%20year%20cyberseurity%20analysis%20%20report%202021.pdf. 2021 年 7 月.)

[3] Ye Y F, Li T, Adjeroh D, et al. A Survey on Malware Detection Using Data Mining Techniques[J]. *ACM Computing Surveys*, 2017, 50(3): 41.

[4] Fu J W, Xue J F, Wang Y, et al. Malware Visualization for Fine-Grained Classification[J]. *IEEE Access*, 2018, 6: 14510-14523.

[5] Pierazzi F, Pendlebury F, Cortellazzi J, et al. Intriguing Properties of Adversarial ML Attacks in the Problem Space[C]. *2020 IEEE Symposium on Security and Privacy*, 2020: 1332-1349.

[6] Raff E, Barker J, Sylvester J, et al. Malware Detection by Eating a Whole EXE[EB/OL]. 2017: 1710.09435.http://arxiv.org/abs/1710.09435v1.

[7] Fan Y J, Hou S F, Zhang Y M, et al. Gotcha - Sly Malware!: Scorpion a Metagraph2vec Based Malware Detection System[C]. *The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018: 253-262.

[8] A. Petitet, R. C. Whaley, J. Dongarra, et al. HPL-a portable implementation of the high-performance linpack benchmark for distributed-memory computers, Version 2.3. https://www.netlib.org/benchmark/hpl/. Dec, 2018.

[9] TPC download current specs/source. https://www.tpc.org/tpc _documents_current_versions/current_specifications5.asp. Oct, 2023.

[10] SPEC's benchmarks and tools. https://www.spec.org/benchmarks .html. Jul, 2023.

[11] Zhang J X, Qin Z, Yin H, et al. A Feature-Hybrid Malware Variants Detection Using CNN Based Opcode Embedding and BPNN Based API Embedding[J]. *Computers & Security*, 2019, 84: 376-392.

[12] Zhang G H, Gao T J, Chen Z G, et al. Study on Malware Classification Based on N-Gram Static Analysis Technology[J]. *Computer Science*, 2022, 49(8): 336-343.
(张光华, 高天娇, 陈振国, 等. 基于 N-Gram 静态分析技术的恶意软件分类研究[J]. *计算机科学*, 2022, 49(8): 336-343.)

[13] Wu X W, Wang Y, Fang Y, et al. Embedding Vector Generation Based on Function Call Graph for Effective Malware Detection and Classification[J]. *Neural Computing and Applications*, 2022, 34(11): 8643-8656.

[14] Islam R, Tian R H, Batten L M, et al. Classification of Malware Based on Integrated Static and Dynamic Features[J]. *Journal of Network and Computer Applications*, 2013, 36(2): 646-656.

[15] Abusnaina A, Khormali A, Alasmary H, et al. Adversarial Learning Attacks on Graph-Based IoT Malware Detection Systems[C]. *2019 IEEE 39th International Conference on Distributed Computing Systems*, 2019: 1296-1305.

[16] Saed Alrabaee. Efficient, scalable, and accurate program fingerprinting in binary code[D]. Montreal, Quebec, Canada: Concordia University Canada, 2018.

[17] Yan J Q, Yan G H, Jin D. Classifying Malware Represented as Control Flow Graphs Using Deep Graph Convolutional Neural Network[C]. *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2019: 52-63.

[18] Wang Shuai, Zhao Yuran, Liu Gongshen. A hierarchical graph-based neural network for malware classification[C]. *The 28th International Conference on Neural Information Processing*, 2021: 621-633.

[19] Ling X, Wu L F, Deng W, et al. MalGraph: Hierarchical Graph

Neural Networks for Robust Windows Malware Detection[C]. *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, 2022: 1998-2007.

[20] Somanna S, Vikhyath Shetty N, Mohammed Afthab M, et al. Classifying Malware Represented as Assembly and Control Flow Graphs Using Ensemble Learning[C]. *Advances in Distributed Computing and Machine Learning*, 2022: 327-337.

[21] Alam S, Sogukpinar I. DroidClone: Attack of the Android Malware Clones - a Step towards Stopping Them[J]. *Computer Science and Information Systems*, 2021, 18(1): 67-91.

[22] Yue Z W, Fang Y, Zhang L. Android Malware Detection Based on Graph Attention Networks[J]. *Journal of Sichuan University* (*Natural Science Edition*), 2022, 59(5): 88-95.
(岳子巍, 方勇, 张磊. 基于图注意力网络的安卓恶意软件检测[J]. *四川大学学报(自然科学版)*, 2022, 59(5): 88-95.)

[23] Zheng R, Wang Q Y, He J, et al. Cryptocurrency Mining Malware Detection Based on Behavior Pattern and Graph Neural Network[J]. *Security and Communication Networks*, 2022, 2022: 9453797.

[24] Zhang Y, Hao J B. Malicious Code Detection Method Based on Attention Mechanism and Residual Network[J]. *Journal of Computer Applications*, 2022, 42(6): 1708-1715.
(张杨, 郝江波. 基于注意力机制和残差网络的恶意代码检测方法[J]. *计算机应用*, 2022, 42(6): 1708-1715.)

[25] Zou Binghui, Cao Chunjie, Wang Longjuan, et al. DACN: malware classification based on dynamic analysis and capsule networks[C]. *Frontiers in Cyber Security*, 2021: 3-13.

[26] Li C, Lv Q J, Li N, et al. A Novel Deep Framework for Dynamic Malware Detection Based on API Sequence Intrinsic Features[J]. *Computers & Security*, 2022, 116: 102686.

[27] Li C, Zheng J J. API Call-Based Malware Classification Using Recurrent Neural Networks[J]. *Journal of Cyber Security and Mobility*, 2021: 617-640.

[28] de Oliveira A S, Jos´e Sassi R. Behavioral Malware Detection Using Deep Graph Convolutional Neural Networks[J]. *International Journal of Computer Applications*, 2021, 174(29): 1-8.

[29] Chen J J, Peng B Z, Wu P Z. Malicious Code Detection Method Based on Dynamic Behavior and Machine Learning[J]. *Computer Engineering*, 2021, 47(3): 166-173.
(陈佳捷, 彭伯庄, 吴佩泽. 基于动态行为和机器学习的恶意代码检测方法[J]. *计算机工程*, 2021, 47(3): 166-173.)

[30] Nguyen H N, Abri F, Pham V, et al. MalView: Interactive Visual Analytics for Comprehending Malware Behavior[J]. *IEEE Access*, 2022, 10: 99909-99930.

[31] Gibert D, Mateu C, Planes J. HYDRA: A Multimodal Deep Learning Framework for Malware Classification[J]. *Computers & Security*, 2020, 95: 101873.

[32] Yang W, Gao M Z, Jiang T. A Malicious Code Static Detection Framework Based on Multi-Feature Ensemble Learning[J]. *Journal of Computer Research and Development*, 2021, 58(5): 1021-1034.
(杨望, 高明哲, 蒋婷. 一种基于多特征集成学习的恶意代码静态检测框架[J]. *计算机研究与发展*, 2021, 58(5): 1021-1034.)

[33] Hojjat Aghakhani, Fabio Gritti, Francesco Mecca, et al. When malware is packin'heat; limits of machine learning classifiers based on static analysis features[C]. *The 27th Annual Network and Distributed System Security Symposium*, 2020.

[34] Yoo S, Kim S, Kim S, et al. AI-HydRa: Advanced Hybrid Approach Using Random Forest and Deep Learning for Malware Classification[J]. *Information Sciences*, 2021, 546: 420-435.

[35] Chaganti R, Ravi V, Pham T D. A Multi-View Feature Fusion Approach for Effective Malware Classification Using Deep Learning[J]. *Journal of Information Security and Applications*, 2023, 72: 103402.

[36] Wang Z Y, Wang W Z, Yang Y Q, et al. CNN‐ and GAN‐Based Classification of Malicious Code Families: A Code Visualization Approach[J]. *International Journal of Intelligent Systems*, 2022, 37(12): 12472-12489.

[37] Chen Z G, Xing S S, Ren X Y. Efficient Windows Malware Identification and Classification Scheme for Plant Protection Information Systems[J]. *Frontiers in Plant Science*, 2023, 14: 1123696.

[38] Asam M, Khan S H, Akbar A, et al. IoT Malware Detection Architecture Using a Novel Channel Boosted and Squeezed CNN[J]. *Scientific Reports*, 2022, 12(1): 15498.

[39] Wang S, Wang J, Song Y F, et al. Malware Variants Detection Model Based on MFF–HDBA[J]. *Applied Sciences*, 2022, 12(19): 9593.

[40] Rosenberg I, Shabtai A, Elovici Y, et al. Query-Efficient Black-Box Attack Against Sequence-Based Malware Classifiers[C]. *The 36th Annual Computer Security Applications Conference*, 2020: 611-626.

[41] Suciu O, Coull S E, Johns J. Exploring Adversarial Examples in Malware Detection[C]. *2019 IEEE Security and Privacy Workshops*, 2019: 8-14.

[42] Kolosnjaji B, Demontis A, Biggio B, et al. Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables[C]. *2018 26th European Signal Processing Conference*, 2018: 533-537.

[43] Hammad M, Garcia J, Malek S. A Large-Scale Empirical Study on the Effects of Code Obfuscations on Android Apps and Anti- Malware Products[C]. *The 40th International Conference on Software Engineering*, 2018: 421-431.

[44] Kirat D, Vigna G. MalGene: Automatic Extraction of Malware Analysis Evasion Signature[C]. *The 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015: 769-780.

[45] Fleshman W, Raff E, Zak R, et al. Static Malware Detection & Subterfuge: Quantifying the Robustness of Machine Learning and Current Anti-Virus[C]. *2018 13th International Conference on Malicious and Unwanted Software*, 2018: 1-10.

[46] Suarez-Tangil G, Dash S K, Ahmadi M, et al. DroidSieve: Fast and Accurate Classification of Obfuscated Android Malware[C]. *The Seventh ACM on Conference on Data and Application Security and Privacy*, 2017: 309-320.

[47] Chen L W, Hou S F, Ye Y F. SecureDroid: Enhancing Security of Machine Learning-Based Detection Against Adversarial Android Malware Attacks[C]. *The 33rd Annual Computer Security Applications Conference*, 2017: 362-372.

[48] Zhan D Z, Hu Y, Li W L, et al. Towards Robust CNN-Based Malware Classifiers Using Adversarial Examples Generated Based on Two Saliency Similarities[J]. *Neural Computing and Applications*, 2023, 35(23): 17129-17146.

[49] Singhal R, Soni M, Bhatt S, et al. Enhancing Robustness of Malware Detection Model Against White Box Adversarial Attacks[C]. *International Conference on Distributed Computing and Intelligent Technology*, 2023: 181-196.

[50] Prajapati P, Stamp M. An Empirical Analysis of Image-Based Learning Techniques for Malware Classification[M]. Malware Analysis Using Artificial Intelligence and Deep Learning. Cham: Springer, 2021: 411-435.

[51] Pinhero A, M L A, Vinod P, et al. Malware Detection Employed by Visualization and Deep Neural Network[J]. *Computers & Security*, 2021, 105: 102247.

[52] Yu M, Jiang J G, Li G, et al. A Survey of Research on Malicious Document Detection[J]. *Journal of Cyber Security*, 2021, 6(3): 54-76.
(喻民, 姜建国, 李罡, 等. 恶意文档检测研究综述[J]. *信息安全学报*, 2021, 6(3): 54-76.)

[53] Afianian A, Niksefat S, Sadeghiyan B, et al. Malware Dynamic Analysis Evasion Techniques: A Survey[J]. *ACM Computing Surveys*, 2019, 52(6): 126.

[54] Abusnaina A, Anwar A, Alshamrani S, et al. Systematically Evaluating the Robustness of ML-Based IoT Malware Detection Systems[C]. *The 25th International Symposium on Research in Attacks, Intrusions and Defenses*, 2022: 308-320.

[55] Ma Y X, Liu S, Jiang J J, et al. A Comprehensive Study on Learning-Based PE Malware Family Classification Methods[C]. *The 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021: 1314-1325.

[56] Park D, Yener B. A Survey on Practical Adversarial Examples for Malware Classifiers[C]. *ROOTS'20: Reversing and Offensive- oriented Trends Symposium*, 2020: 23-35.

[57] Wagner M, Fischer F, Luh R, et al. A Survey of Visualization Systems for Malware Analysis[J]. *Eurographics Conference on Visualization - State of the Art Reports, EuroVis-STAR 2015*, 2015: 105-125.

[58] Ling X, Wu L F, Zhang J Y, et al. Adversarial Attacks Against Windows PE Malware Detection: A Survey of the State-of-the-Art[J]. *Computers & Security*, 2023, 128: 103134.

[59] Nataraj L, Karthikeyan S, Jacob G, et al. Malware Images: Visualization and Automatic Classification[C]. *The 8th International Symposium on Visualization for Cyber Security*, 2011: 1-7.

[60] Conti G, Dean E, Sinda M, et al. Visual Reverse Engineering of Binary and Data Files[C]. *The 5th international workshop on Visualization for Computer Security*, 2008: 1-17.

[61] Kancherla K, Mukkamala S. Image Visualization Based Malware Detection[C]. *2013 IEEE Symposium on Computational Intelligence in Cyber Security*, 2013: 40-44.

[62] Kancherla K, Donahue J, Mukkamala S. Packer Identification Using Byte Plot and Markov Plot[J]. *Journal of Computer Virology and Hacking Techniques*, 2016, 12(2): 101-111.

[63] Hashemi H, Hamzeh A. Visual Malware Detection Using Local Malicious Pattern[J]. *Journal of Computer Virology and Hacking Techniques*, 2019, 15(1): 1-14.

[64] Luo J S, Lo D C T. Binary Malware Image Classification Using Machine Learning with Local Binary Pattern[C]. *2017 IEEE International Conference on Big Data*, 2017: 4664-4667.

[65] Le Q, Boydell O, Mac Namee B, et al. Deep Learning at the Shallow End: Malware Classification for Non-Domain Experts[J]. *Digital Investigation*, 2018, 26: S118-S126.

[66] Vinayakumar R, Alazab M, Soman K P, et al. Robust Intelligent Malware Detection Using Deep Learning[J]. *IEEE Access*, 2019, 7: 46717-46738.

[67] Venkatraman S, Alazab M, Vinayakumar R. A Hybrid Deep Learning Image-Based Analysis for Effective Malware Detection[J]. *Journal of Information Security and Applications*, 2019, 47: 377-389.

[68] Saridou B, Moulas I, Shiaeles S, et al. Image-Based Malware Detection Using A-Cuts and Binary Visualisation[J]. *Applied Sciences*, 2023, 13(7): 4624.

[69] Khan R U, Zhang X S, Kumar R. Analysis of ResNet and GoogleNet Models for Malware Detection[J]. *Journal of Computer Virology and Hacking Techniques*, 2019, 15(1): 29-37.

[70] Qiao Y C, Jiang Q S, Jiang Z C, et al. A Multi-Channel Visualization Method for Malware Classification Based on Deep Learning[C]. *2019 18th IEEE International Conference on Trust, Security and Privacy In Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering*, 2019: 757-762.

[71] Zhang J X, Qin Z, Yin H, et al. IRMD: Malware Variant Detection Using Opcode Image Recognition[C]. *2016 IEEE 22nd International Conference on Parallel and Distributed Systems*, 2016: 1175-1180.

[72] Ni S, Qian Q, Zhang R. Malware Identification Using Visualization Images and Deep Learning[J]. *Computers & Security*, 2018, 77: 871-885.

[73] Han K, Lim J H, Im E G. Malware Analysis Method Using Visualization of Binary Files[C]. *The 2013 Research in Adaptive and Convergent Systems*, 2013: 317-321.

[74] Xiao G Q, Li J N, Chen Y D, et al. MalFCS: An Effective Malware Classification Framework with Automated Feature Extraction Based on Deep Convolutional Neural Networks[J]. *Journal of Parallel and Distributed Computing*, 2020, 141: 49-58.

[75] Nguyen H, Di Troia F, Ishigaki G, et al. Generative Adversarial Networks and Image-Based Malware Classification[J]. *Journal of Computer Virology and Hacking Techniques*, 2023, 19(4): 579-595.

[76] Ganesh M, Pednekar P, Prabhuswamy P, et al. CNN-Based Android Malware Detection[C]. *2017 International Conference on Software Security and Assurance*, 2017: 60-65.

[77] D'Angelo G, Ficco M, Palmieri F. Malware Detection in Mobile Environments Based on Autoencoders and API-Images[J]. *Journal of Parallel and Distributed Computing*, 2020, 137: 26-33.

[78] Yuan B G, Wang J F, Liu D, et al. Byte-Level Malware Classification Based on Markov Images and Deep Learning[J]. *Computers & Security*, 2020, 92: 101740.

[79] Zhao Z L, Zhao D W, Yang S M, et al. Image-Based Malware Classification Method with the AlexNet Convolutional Neural Network Model[J]. *Security and Communication Networks*, 2023, 2023: 6390023.

[80] O'Shaughnessy S, Sheridan S. Image-Based Malware Classification Hybrid Framework Based on Space-Filling Curves[J]. *Computers & Security*, 2022, 116: 102660.

[81] Deng H X, Guo C, Shen G W, et al. MCTVD: A Malware Classification Method Based on Three-Channel Visualization and Deep Learning[J]. *Computers & Security*, 2023, 126: 103084.

[82] Conti M, Khandhar S, Vinod P. A Few-Shot Malware Classification Approach for Unknown Family Recognition Using Malware Feature Visualization[J]. *Computers & Security*, 2022, 122: 102887.

[83] Huang T H D, Kao H Y. R2-D2: ColoR-Inspired Convolutional NeuRal Network (CNN)-Based AndroiD Malware Detections[C]. *2018 IEEE International Conference on Big Data*, 2018: 2633-2642.

[84] Vasan D, Alazab M, Wassan S, et al. IMCFN: Image-Based Malware Classification Using Fine-Tuned Convolutional Neural Network Architecture[J]. *Computer Networks*, 2020, 171: 107138.

[85] Qi X Y, Liu W, Lou R, et al. MC-ISA: A Multi-Channel Code Visualization Method for Malware Detection[J]. *Electronics*, 2023, 12(10): 2272.

[86] Xiao M, Guo C, Shen G W, et al. Image-Based Malware Classification Using Section Distribution Information[J]. *Computers & Security*, 2021, 110: 102420.

[87] Yen Y S, Sun H M. An Android Mutation Malware Detection Based on Deep Learning Using Visualization of Importance from Codes[J]. *Microelectronics Reliability*, 2019, 93: 109-114.

[88] Moussas V, Andreatos A. Malware Detection Based on Code Visualization and Two-Level Classification[J]. *Information*, 2021, 12(3): 118.

[89] Marastoni N, Giacobazzi R, Dalla Preda M. Data Augmentation and Transfer Learning to Classify Malware Images in a Deep Learning Context[J]. *Journal of Computer Virology and Hacking Techniques*, 2021, 17(4): 279-297.

[90] Go J H, Jan T, Mohanty M, et al. Visualization Approach for Malware Classification with ResNeXt[C]. *2020 IEEE Congress on Evolutionary Computation*, 2020: 1-7.

[91] Vinita Verma, Sunil K.Muttoo, V.B.Singh. Multiclass malware classification via first- and second-order texture statistics [J]. *Computers and Security*, 2020, 97: 101895.

[92] Niket Bhodia, Pratikkumar Prajapati, Fabio Di Troia, et al. Transfer learning for image-based malware classification[C]. *The 3rd International Workshop on Formal Methods for Security Engineering*, 2019: 1-7.

[93] Verma V, Muttoo S K, Singh V □. Multiclass Malware Classification via First- and Second-Order Texture Statistics[J]. *Computers & Security*, 2020, 97: 101895.

[94] Naeem H, Guo B, Naeem M R, et al. Identification of Malicious Code Variants Based on Image Visualization[J]. *Computers & Electrical Engineering*, 2019, 76: 225-237.

[95] Gibert D, Mateu C, Planes J, et al. Using Convolutional Neural Networks for Classification of Malware Represented as Images[J]. *Journal of Computer Virology and Hacking Techniques*, 2019, 15(1): 15-28.

[96] Cui Z H, Xue F, Cai X J, et al. Detection of Malicious Code Variants Based on Deep Learning[J]. *IEEE Transactions on Industrial Informatics*, 2018, 14(7): 3187-3196.

[97] Kalash M, Rochan M, Mohammed N, et al. Malware Classification with Deep Convolutional Neural Networks[C]. *2018 9th IFIP International Conference on New Technologies*, *Mobility and Security*, 2018: 1-5.

[98] Makandar A, Patrot A. Malware Analysis and Classification Using Artificial Neural Network[C]. *2015 International Conference on Trends in Automation*, *Communications and Computing Technology*, 2015: 1-6.

[99] Han K S, Lim J H, Kang B, et al. Malware Analysis Using Visualized Images and Entropy Graphs[J]. *International Journal of Information Security*, 2015, 14(1): 1-14.

[100] Yan J P, Qi Y, Rao Q F. Detecting Malware with an Ensemble Method Based on Deep Neural Network[J]. *Security and Communication Networks*, 2018, 2018: 7247095.

[101] Li Chen, Ravi Sahita, Jugal Parikh, et al. STAMINA: scalable deep learning approach for malware classification. https://www.intel.com/content/dam/www/public/us/en/ai/documents/stamina-scalable-deep-learning-whitepaper.pdf. Apr 2020.

[102] Bakour K, Ünver H M. DeepVisDroid: Android Malware Detection by Hybridizing Image-Based Features with Deep Learning Techniques[J]. *Neural Computing and Applications*, 2021, 33(18): 11499-11516.

[103] Luo Shiqi, Tian Shengwei, Yu long, et al. Android malicious code classification using deep belief network[J]. *KSII Transactions on Internet and Information Systems*, 2018, 12(1): 454-475.

[104] Zhong F T, Chen Z K, Xu M H, et al. Malware-on-the-Brain: Illuminating Malware Byte Codes with Images for Malware Classification[J]. *IEEE Transactions on Computers*, 2023, 72(2): 438-451.

[105] Liu X B, Lin Y P, Li H, et al. A Novel Method for Malware Detection on ML-Based Visualization Technique[J]. *Computers & Security*, 2020, 89: 101682.

[106] Darem A, Abawajy J, Makkar A, et al. Visualization and Deep-Learning-Based Malware Variant Detection Using Op-Code-Level Features[J]. *Future Generation Computer Systems*, 2021, 125: 314-323.

[107] Liu Y S, Wang Z H, Yan H B, et al. Method of Anti-Confusion Texture Feature Descriptor for Malware Images[J]. *Journal on Communications*, 2018, 39(11): 44-53.
(刘亚姝, 王志海, 严寒冰, 等. 抗混淆的恶意代码图像纹理特征描述方法[J]. *通信学报*, 2018, 39(11): 44-53.)

[108] Zhang C B, Zhang Y C, Zheng Y, et al. Malware Classification Based on Texture Fingerprint of Gray-Scale Images[J]. *Computer Science*, 2018, 45(S1): 383-386.
(张晨斌, 张云春, 郑杨, 等. 基于灰度图纹理指纹的恶意软件分类[J]. *计算机科学*, 2018, 45(S1): 383-386.)

[109] Jiang K L, Bai W, Zhang L, et al. Malicious Code Detection Based on Multi-Channel Image Deep Learning[J]. *Journal of Computer Applications*, 2021, 41(4): 1142-1147.
(蒋考林, 白玮, 张磊, 等. 基于多通道图像深度学习的恶意代码检测[J]. *计算机应用*, 2021, 41(4): 1142-1147.)

[110] Gulmez S, Kakisim A G, Sogukpinar I. Analysis of the Dynamic Features on Ransomware Detection Using Deep Learning-Based Methods[C]. *2023 11th International Symposium on Digital Forensics and Security*, 2023: 1-6.

[111] Komatwar R, Kokare M. Conglomerate Stratum Model for Categorization of Malware Family in Image Processing[J]. *Fuzzy Information and Engineering*, 2023, 15(3): 203-219.

[112] Ullah F, Alsirhani A, Alshahrani M M, et al. Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation[J]. *Sensors*, 2022, 22(18): 6766.

[113] Van Dao T, Sato H, Kubo M S. MLP-Mixer-Autoencoder: A Lightweight Ensemble Architecture for Malware Classification[J]. *Information*, 2023, 14(3): 167.

[114] Katrina Tran, Fabio Di Troia, Mark Stamp. Robustness of image-based malware analysis[C]. *Silicon Valley Cybersecurity Conference*, 2022: 3-21.

[115] Ahmed I T, Jamil N, Din M M, et al. Binary and Multi-Class Malware Threads Classification[J]. *Applied Sciences*, 2022, 12(24): 12528.

[116] Zhao Y, Kuerban A. MDABP: A Novel Approach to Detect Cross-Architecture IoT Malware Based on PaaS[J]. *Sensors*, 2023, 23(6): 3060.

[117] Singh P, Priyanshu, Bhat A. Malware Classification Using Malware Visualization and Deep Learning[C]. *2023 2nd International Conference on Applied Artificial Intelligence and Computing*, 2023: 528-534.

[118] Khan S H, Alahmadi T J, Ullah W, et al. A New Deep Boosted CNN and Ensemble Learning Based IoT Malware Detection[J]. *Computers & Security*, 2023, 133: 103385.

[119] Kyoung-Won Park, Sung-Bae Cho. A vision Transformer enhanced with patch encoding for malware classification[C]. *International Conference on Intelligent Data Engineering and Automated Learning*, 2022 289–299.

[120] Li Y M, Li C H, Song Y F, et al. Method of Malware Family Classification Based on Attention-DenseNet-BC Model Mechanism[J]. *Computer Science*, 2021, 48(10): 308-314.
(李一萌, 李成海, 宋亚飞, 等. 基于 Attention-DenseNet-BC 的恶意软件家族分类方法[J]. *计算机科学*, 2021, 48(10): 308-314.)

[121] Shao Y L, Lu Y, Wei D, et al. Malicious Code Classification Method Based on Deep Residual Network and Hybrid Attention Mechanism for Edge Security[J]. *Wireless Communications and Mobile Computing*, 2022, 2022: 3301718.

[122] Chong X L, Gao Y T, Zhang R, et al. Classification of Malware Families Based on Efficient-Net and 1D-CNN Fusion[J]. *Electronics*, 2022, 11(19): 3064.

[123] Belal M M, Sundaram D M. Global-Local Attention-Based Butterfly Vision Transformer for Visualization-Based Malware Classification[J]. *IEEE Access*, 2023, 11: 69337-69355.

[124] Osho Sharma, Akashdeep Sharma, Arvind Kalia. Windows malware hunting with InceptionResNetv2 assisted malware visualization approach[C]. *International Conference on Computational In-*

telligence and Data Engineering, 2022: 171–188.

[125] Chaganti R, Ravi V, Pham T D. Image-Based Malware Representation Approach with EfficientNet Convolutional Neural Networks for Effective Malware Classification[J]. *Journal of Information Security and Applications*, 2022, 69: 103306.

[126] Panda P, C U O K, Marappan S, et al. Transfer Learning for Image-Based Malware Detection for IoT[J]. *Sensors*, 2023, 23(6): 3253.

[127] Singh A, Dutta D, Saha A. MIGAN: Malware Image Synthesis Using GANs[J]. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, 33(1): 10033-10034.

[128] Qiu H, Zeng Y, Zhang T W, et al. FenceBox: A Platform for Defeating Adversarial Examples with Data Augmentation Techniques [EB/OL]. 2020: 2012.01701.http://arxiv.org/abs/2012.01701v1.

[129] Shoumik Saha, Sadia Afroz, Atif Rahman. MALIGN: adversarially robust malware family detection using sequence alignment. https://arxiv.org/abs/2111.14185v1. Nov 2021.

[130] Freitas S, Duggal R, Chau D H. MalNet: A Large-Scale Image Database of Malicious Software[C]. *The 31st ACM International Conference on Information & Knowledge Management*, 2022: 3948-3952.

钱丽萍 于 2015 年在中国人民大学计算机应用专业获得博士学位。现任北京建筑大学电信学院副教授。研究领域为智能信息处理、网络安全。当前研究兴趣包括: 恶意软件对抗生成、网络安全开源情报分析。Email: qianliping@bucea.edu.cn

王大伟 于 2010 年在哈尔滨理工大学计算机应用专业获得博士学位。现任国家计算机网络应急技术处理协调中心正高级工程师。研究领域为信息安全、网络流量检测。当前研究兴趣包括: 异常流量分析、恶意模式挖掘。Email: stonetools@yeah.net