计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2021.20200964
58(5): 977-994, 2021

Windows平台恶意软件智能检测综述

汪嘉来[1,2] 张 超[1,2] 戚旭衍[3] 荣 易[4]
[1](清华大学网络科学与网络空间研究院 北京 100084)
[2](北京信息科学与技术国家研究中心 北京 100084)
[3](数学工程与先进计算国家重点实验室 郑州 450002)
[4](清华大学软件学院 北京 100084)
(wangjl19@mails.tsinghua.edu.cn)

A Survey of Intelligent Malware Detection on Windows Platform

Wang Jialai[1,2], Zhang Chao[1,2], Qi Xuyan[3], and Rong Yi[4]
[1](*Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing* 100084)
[2](*Beijing National Research Center for Information Science and Technology, Beijing* 100084)
[3](*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou* 450002)
[4](*School of Software, Tsinghua University, Beijing* 100084)

Abstract In recent years, malware has brought many negative effects to the development of information technology. In order to solve this problem, how to effectively detect malware has always been a concern. With the rapid development of artificial intelligence, machine learning and deep learning technologies are gradually introduced into the field of malware detection. This type of technology is called intelligent malware detection technology. Compared with traditional detection methods, intelligent detection technology does not need to manually formulate detection rules due to the application of artificial intelligence technology. Besides, intelligent detection technology has stronger generalization capabilities, and can better detect previously unseen malware. Intelligent malware detection has become a research hotspot in the field of detection. This paper mainly introduces current work related to intelligent malware detection, which includes the main parts required for intelligent detection processes. Specifically, we have systematically explained and classified related work for intelligent malware detection in this paper, which includes the features commonly used in intelligent detection, how to perform feature processing, the commonly used classifiers in intelligent detection, and the main problems faced by current malware intelligent detection. Finally, we summarize the full paper and clarify the potential future research directions, aiming to contribute to the development of intelligent malware detection.

Key words malware; intelligent malware detection; artificial intelligence; machine learning; deep learning

摘 要 近年来,恶意软件给信息技术的发展带来了很多负面的影响.为了解决这一问题,如何有效检测恶意软件则一直备受关注.随着人工智能的迅速发展,机器学习与深度学习技术逐渐被引入到恶意软件

收稿日期:2020-11-19;**修回日期**:2021-02-25
基金项目:国家自然科学基金面上项目(61972224)
This work was supported by the General Program of the National Natural Science Foundation of China (61972224).
通信作者:张超(chaoz@tsinghua.edu.cn)

的检测中,这类技术称之为恶意软件智能检测技术.相比于传统的检测方法,由于人工智能技术的应用,智能检测技术不需要人工制定检测规则.此外,具有更强的泛化能力,能够更好地检测先前未见过的恶意软件.恶意软件智能检测已经成为当前检测领域的研究热点.主要介绍了当前的恶意软件智能检测相关工作,包含了智能检测所需的主要环节.从智能检测中常用的特征、如何进行特征处理、智能检测中常用的分类器、当前恶意软件智能检测所面临的主要问题 4 个方面对智能检测相关工作进行了系统地阐述与分类.最后,总结了先前智能检测相关工作,阐明了未来潜在的研究方向,旨在能够助力恶意软件智能检测的发展.

关键词 恶意软件;恶意软件智能检测;人工智能;机器学习;深度学习

中图法分类号 TP309

随着社会的发展,计算机系统已经成为人们日常生活中不可或缺的一部分.而恶意软件的存在,则一直威胁着计算机系统的安全,给社会的发展带来了巨大的损失.例如 2014 年全球遭受的与恶意软件相关的经济损失大约为 5 000 亿美元[1].对此,如何有效检测恶意软件一直是研究的热点.

传统的恶意软件检测技术往往需要大量的人工参与.如基于签名的检测技术、启发式的检测技术等,它们需要人工制定许多检测规则[2].然而,这类传统的检测技术无法很好地检测先前未见过的新式恶意软件[3].尤其是在恶意软件日均新增数量众多的情况下,该类技术更加无法胜任.近些年来,随着人工智能的蓬勃发展,恶意软件的检测技术也开始与机器学习、深度学习技术相结合,我们称之为恶意软件智能检测技术.这种新的检测技术能够更好地检测先前未见过的新式样本,更能够满足当前社会的安全需求.作为新型恶意软件检测技术的代表,恶意软件智能检测技术具有泛化能力强的特点,还能够在一定程度上降低人工的参与.因此,研究有效的恶意软件智能检测技术具有较高的价值.

对于恶意软件智能检测相关研究而言,如何有效应用人工智能技术是其中的关键.从宏观上看,应用人工智能技术需要包含 3 个主要步骤:特征提取、特征处理和分类器设计.而这 3 个主要步骤,也是当前恶意软件智能检测相关工作的主要研究点.在面对恶意软件智能检测场景时,相关工作需要结合恶意软件的特点,有针对性的设计这 3 个主要环节.同时,由于是智能检测,则意味着机器学习与深度学习技术所面临的一些问题也必须被充分考虑.如收集数据集、避免过拟合、提升训练速度等.

本文旨在对当前恶意软件智能检测相关工作进行总结与概括.恶意软件智能检测架构如图 1 所示:

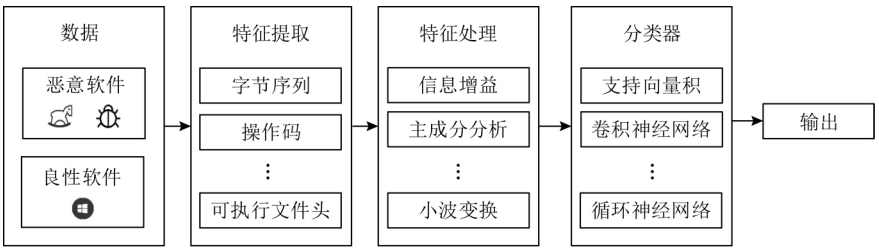

Fig. 1 The framework of malware intelligent detection

图 1 恶意软件智能检测架构

对于恶意软件智能检测技术相关工作而言,其研究点较为明确,而本文也将根据这些研究点来进行介绍.具体而言,本文将主要从恶意软件智能检测常用特征、特征处理、智能检测常用分类器以及现阶段智能检测主要面临的问题这 4 个方面来进行阐述.另外,由于 Windows 系统是当前使用最广泛的操作系统之一,而针对 Windows 系统的恶意软件也属于恶意软件的主流.对此,本文主要介绍针对 Windows 平台上的恶意软件智能检测相关工作.

1 常用特征

常用特征的制定是智能检测的重要环节.其重要性主要体现于:作为分类器的输入,常用特征代表

了待测程序,它决定了分类器能否学习到程序的本质.由于计算机软件具有数量庞大、语义信息丰富、结构复杂的特点,现阶段大部分工作还不能完全依靠深度学习技术来自动化提取待测程序特征.对此,人工参与常用特征制定依旧是当前恶意软件智能检测技术的主流.

本节主要介绍恶意软件智能检测相关工作的常用特征.对于这些常用特征而言,获取方式并不完全相同:有些特征需要静态获取,有些需要动态获取,还有些特征则可以同时通过动、静态方法获取.在本节中,将按照静态、动态、动静态获取的顺序依次展开介绍.

1.1 静态特征

本节主要介绍待测程序的静态特征.静态特征的提取不需要运行待测程序,它是直接从待测的二进制程序中提取需要的数据.在提取的过程中,只需要确定相应数据的位置及提取的方式即可,如:1)直接按照一定规则切分程序字节,获取字节序列特征;2)利用现成的工具提取程序中的可阅读字符串;3)定位待测程序头部,获取文件头部信息;4)按照需求,计算程序文件相应部分熵值;5)定位静态文件中动态链接库(dynamic link library, DLL)描述信息位置,提取动态链接库相应信息;6)按照一定规则,直接将待测程序数据转换为图片的形式.

1.1.1 字节序列

字节序列是字节级的程序特征.对于程序而言,每个字节本身都包含了信息.因此,可以通过不同的方式提取字节,构建字节序列,进而得到不同的程序特征.

按照一定规则切分待测程序字节是得到程序特征的直接方式.Schultz 等人[4]首先将程序中所有的字节转化为对应的十六进制数据形式,随后再按规定字节数划分字节序列,划分的结果即为多组字节序列.这些字节序列将作为特征使用.对于每组字节序列而言,本质上都能够对应一个指令序列.这种方式获取的程序特征较为简单直观.

使用 n-gram 方法是另一种提取字节序列的方式,提取方式如图 2 所示.该方法是对程序字节进行大小为 n 的滑动窗口操作,形成多个长度为 n 的字节序列;而每种序列出现的频率即可作为该程序的特征.Schultz 等人[4]是首个在 Windows 平台上使用 n-gram 方法来提取特征的团队.通过将提取的特征放在不同的分类器下进行尝试,并经过横向对比,他们发现 n-gram 方法比单纯基于签名进行检测的方法更好.Kolter 等人[5]则专精于使用 n-gram 方法进行特征提取,并尝试了多个不同的 n 值,旨在其恶意软件检测实验中获得最优效果.另一方面,n-gram 在字节序列上提取的特征还可与其他特征进行结合使用[6-8].例如,与操作码上的 n-gram 特征、动态链接库函数导入特征相结合等.就提取字节序列特征而言,Raff 等人[9]对众多使用 n-gram 方法进行特征提取的恶意软件检测工作进行了系统的评价,并揭示了许多问题,如高估准确率、提取特征方法较为冗余、过拟合等.

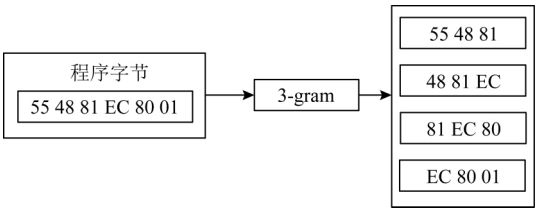

Fig. 2 "3-gram"byte sequences

图 2 "3-gram"字节序列

2 种方法本质上都是对程序字节进行不同方式的切分,以此来获得不同的字节序列,进而作为程序的特征.然而,Raff 等人[10]提出了一种不需要切分程序字节,而是将整个程序字节作为字节序列送入分类器的方法.他们直接通过嵌入(embedding),将程序字节转换为固定长度的向量,并以此来构建字节之间的语义关系.以此方法获得的向量将被送入神经网络中进一步提取特征并最终用于恶意软件检测.

1.1.2 可阅读字符串

对于某些恶意程序而言,它们会包含一些固定的可阅读字符串.如字符串"〈html〉〈script language='javascript'〉window.open ('readme.eml')",其经常出现在"Nimda"蠕虫中[11].尽管恶意软件可能会不断改变自身的静态特征来躲避检查,但它很难让原本包含的可阅读字符串全部发生改变.因此,对于待检测程序中的可阅读字符串,也可以作为智能检测特征.

对于提取可阅读字符串,可以直接使用现成的工具进行提取,如"Linux 下的 strings 命令"[4];也可以自行设计一些方法进行提取.如 Ye 等人[11],通过遍历程序文件,首先查看是否有连续的字节属于同一字符集(如查看是否有连续的字节属于 ASCII 字符集).如果有,则将相应的连续字节保存下来;随后再利用先前准备好的自然语言集合对其进一步筛选,判断该连续字节是否为可阅读字符串.

1.1.3 文件头部信息

文件头部信息也可以作为程序特征.由于本文讨论的被检测对象是 Windows 平台下的可运行程序(portable executable, PE)格式,故在此以 PE 文件的文件头部为例.PE 文件头部包含了被检测程序的众多信息,包括程序文件的节数、节名、节大小、时间戳、动态链接库信息、导入表信息、导出表信息等.这些信息可以看作是对待测程序的宏观描述.对此,文件头部信息常作为恶意软件智能检测的重要特征[3,12-13].

具体而言,Shafiq 等人[12]针对文件头部信息进行特征提取时,他们几乎遍历了头部信息中所有可用于提取的字段,提取出来的特征种类高达 189 项.但这些提取出来的特征并不会全部用于智能检测,而是会通过特征工程进行筛选后,再用于检测.

另一方面,文件头部信息不仅可以直接作为特征,还可以被用于衍生出其他特征.Kumar 等人[13]在提取出头部信息后,会选取一部分信息作为程序特征,还会在某些头部信息的基础上,衍生出一些其他的特征.如根据时间戳的合法性定义的布尔变量特征:通过检查头部时间戳是否合法,给出相应的布尔变量值,并以此作为特征.

1.1.4 熵

在信息论中,熵是对不确定性的测量.对于程序而言,熵越高,则代表其中的数据越随机.由于熵能够衡量程序中数据的随机性,恶意软件检测经常会用到熵.例如,对于一些加壳的恶意程序,由于加壳后的数据较为随机,其相应部分的熵也偏高,很多恶意软件检测厂商便可以根据这一特性来进行检测恶意软件.对于恶意软件智能检测,熵可以作为重要特征[13-16].

在提取熵时,不仅可以计算整个文件的熵,还可以针对程序的某些部分计算熵.除了整个程序文件的熵外,Kumar 等人[13]还专门对数据段和代码段计算了熵,并用这 3 种熵值作为特征.除了直接用熵作为特征外,还可以根据熵来衍生其他特征.如 Markel 等人[14]单独设置了一个“HighEntropy”特征,当待测程序的任意一个节的熵值大于 7 时,则该特征被置为 1,反之则为 0,这便是一种依据熵来衍生其他特征的典型方法.

1.1.5 动态链接库相关信息

对于 Windows 平台下的待测程序,直接通过静态解析,便可以获得当前程序所使用的动态链接库、动态链接库中被导入函数等相关信息,这些与动态链接库相关的信息都可以作为程序的特征.

根据程序 DLL 的相关信息,可以在一定程度上推测一个程序可能拥有什么功能、不能执行什么功能[4].Schultz 等人[4]利用了已有的 DLL 相关信息,并对程序行为进行了推测.例如,假如一个待测程序没有使用“USER32.DLL”,则可以推测该待测程序没有使用 Windows 用户接口.不过这种推测方式并不精确,因为待测程序中可能自身编写或链接了其他的用户接口库.虽然这种推测方式并不精确,但能够在一定程度上反应程序功能,所以将动态链接库相关信息来作为特征是可行的.

在提取特征的方法上,Schultz 等人[4]直接从程序头部中提取所有动态链接库函数相关信息,并分别以待测程序使用的动态链接库集合、动态链接库中调用的函数集合、不同动态链接库调用的函数个数来作为特征.在进行特征表示时,他们会把功能相似的动态链接库放在一起.而 Shafiq 等人[12]并没有把相似的动态链接库放在一起,他们通过实验证明单独考虑每一个 DLL 效果更好.他们先总结了 73 个核心动态链接库,随后再针对每一个待测程序,通过布尔变量表示这些动态链接库是否存,进而作为特征.Masud 等人[17]是在反汇编后的文件中提取所有的动态链接库函数,随后再对提取的所有函数使用 n-gram 方法提取函数序列特征.Bai 等人[18]在获取到待测程序所用的 DLL 和 DLL 中的被导入函数后,会进行统计筛选.他们会把出现频率较低的 DLL 筛去,并通过信息增益的方法留下最优的 30 个 DLL 和 30 个 DLL 中记录的被导入函数.除此以外,他们还考虑将 DLL,DLL 中被导入函数的数量以及符号导出表的信息作为智能检测的特征.

1.1.6 神经网络自动提取特征

除了人工制定特征外,还可以运用神经网络来自动提取待测程序特征.在自动提取特征的相关工作中,需要考虑数据的表示形式,便于神经网络能够较好地提取到关键特征,从而保证检测结果准确可靠.

对于待测程序而言,可以将其转换为图片的形式,再交由神经网络提取特征[19-20].此类工作认为程序的结构相似性可以体现在对应图片的相似性上[21].Yan 等人[20]就是将程序数据转换为图的形式,再交由神经网络提取特征.由于程序字节的取值范围与像素值相同,他们直接将程序字节与像素值一一对应,进而构建相应的灰度图,具体例子如图 3

所示(此图为某可执行程序对应的灰度图).由于神经网络的输入格式一般是固定的,所以输入图片的大小也需要统一.因此,对于不同程序的灰度图,需要统一修改至相同的大小.对此,他们使用了双线性插值的方法来调整图片大小.

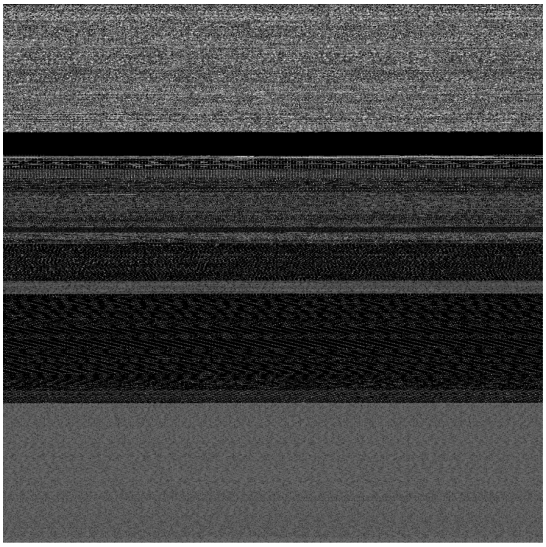

Fig. 3 Grayscale image corresponding to the program

图 3 程序对应的灰度图

此外,还可以将待测程序转化为向量的形式.如 Raff 等人[10]使用 embedding 的方法将程序字节转化为向量,随后再交由神经网络进行检测.他们不赞成直接将程序转换为图的形式.他们认为图片和程序有本质区别,将程序转换为图片的形式从根本上来讲是行不通的.另外,由于图片大小固定,则必然涉及到填充或丢弃某些字节,这会带来 2 点新的问题:1)在填充字节时,意味着程序对应的图片大小不足,需要填充字节来满足格式.然而填充的字节往往不具有实际意义.如运用双线性插值进行填充时,填充的字节由数学运算得到,具备数学意义,但与程序语义没有任何关系.他们认为这样无法帮助智能检测.2)在丢弃字节时,意味着程序对应的图片大小超出了标准格式范围,因此通常会对程序字节采用截断处理.然而,如果程序的恶意行为藏在了程序尾部,则会由于截断的发生而无法被检测.

1.2 动态特征

本节主要介绍待测程序的动态特征.动态特征的提取需要运行待测程序.常用的做法是将待测程序运行在可控的虚拟环境中.为了捕获待测程序的动态行为,首先需要保证程序能够正常运行.即在动态运行程序时,需要尽可能地满足程序所需的运行时环境,如提供合适的操作系统版本,安装待测所需的软件,开放相应的端口等.此外,还需要考虑如何充分展现程序的行为,避免待测程序隐藏功能或行为展现不完全等.如在获取程序动态行为时,需要模拟反馈所需的网络交互信息,或通过强制执行等手段探索某些未被触发的恶意功能等.

对于运行时状态信息而言,需要靠动态方法收集.常用的做法是将待测程序运行在虚拟环境中,进而提取相应的运行时状态信息.运行时状态信息主要包含:CPU、寄存器、IO、线程、文件、内存、网络活动等相关信息[22-24].

通过在沙箱中动态运行待测程序,Burnap 等人[22]自己编写了脚本来获取运行时状态信息,并主要将 CPU 用户使用百分比、CPU 系统使用百分比、RAM 的使用数量、虚拟内存分区的使用数量、进程运行数、发送/收到的网络包数量以及发送/收到的字节数等作为特征.而 Abdelsalam 等人[19]考虑的状态信息更多,比如额外考虑了使用的线程数、打开的文件数、虚拟内存大小、共享库内存占用等.Mohaisen 等人[25]主要以注册表修改操作(如创建、修改键值)、文件系统修改操作(如文件创建、修改、删除)以及网络活动作为特征.

1.3 静、动态特征

除了静态特征与动态特征以外,还有些程序特征既可以通过静态获取,也可以通过动态获取:1)对于操作码序列,在静态条件下,可以通过反汇编二进制程序,并根据反汇编后的指令获取操作码序列;在动态条件下,可以通过动态运行程序,并将内存中的指令读取出来,随后提取操作码序列;2)对于 API 及系统调用,在静态条件下,可以分析二进制程序的控制流图(controw flow graph, CFG),然后根据 CFG 获取相应的 API 信息;在动态条件下,可以在虚拟环境中记录待测程序的 API 及系统调用信息.

1.3.1 操作码序列

对于操作码(opcode)而言,现有的工作多半是统计操作码的频率,或者使用各种方法来提取操作码序列如 n-gram,使用 3-gram 提取的操作码序列如图 4 所示.如果在提取操作码的同时,还提取了操作码的操作数,则可获得反汇编序列.

运用操作码相关特征需要先反汇编待测程序,然后根据反汇编得到的指令,获得所有的操作码.在此基础上,即可统计操作码频率或提取操作码序列(如果同时提取了操作数则为反汇编序列).就操作

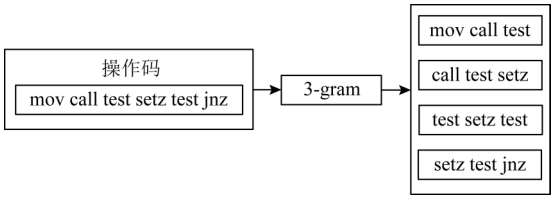

Fig. 4 "3-gram"opcode sequences

图 4 "3-gram"操作码序列

码序列而言,不同于字节序列,后者通常考虑待检测程序的全部字节,而操作码序列只来源于程序中指令所对应的字节.

对于操作码频率而言,Ye 等人[26]是直接以操作码出现的频率来作为特征.而一些其他工作则在操作码频率的基础上做进一步处理.Khodamoradi 等人[27]对操作码的频率统计结果进一步分层,他们划分出了:全部操作码集合、频率非零的操作码集合、高频操作码集合,并以这些集合为基础来构建程序特征.Anderson 等人[28]认为直接使用操作码频率会导致特征空间太大,他们先对提取的操作码进行分类,如按照分支跳转操作、内存操作、堆栈操作等来对操作码进行划分,随后再将分类后的结果作为特征.这样便充分压缩了特征空间.

对于获取的操作码,同样也可以使用 n-gram 方法进行特征提取[17-29].如 Masud 等人[17]针对获取的反汇编语句集合使用 n-gram 方法,并得到相应的反汇编序列,进而作为程序特征.还有一些工作在构造操作码序列时,没有使用 n-gram 方法.Ye 等人[26]使用的是以函数为单位的操作码序列.他们以函数基本块为单位,先从程序中识别出不同的函数,再提取对应的操作码,这样便得到了以函数为单位的操作码序列,随后再以此作为特征.

1.3.2 API 及系统调用

对于 API 及系统调用的获取,可以通过动态或静态的方法收集.就动态方法而言,通常会将恶意软件放入虚拟环境(如沙箱)中执行,并观测记录相应的 API 及系统调用信息.就静态方法而言,通常需要先反汇编待测程序,并提取相应的 CFG(控制流图),然后根据 CFG 获得相应的 API 信息.

Rieck 等人[30]在通过动态分析方法获取程序的 API 后,主要通过统计待测程序各 API 出现的频率来构建特征.而 Qiao 等人[31]则是依据 API 序列的频率来构建特征.Eskandari 等人[32]认为除了获得程序的 API 种类,保留 API 序列的调用顺序信息也是必要的.他们通过对提取的 API 使用 n-gram 的方法获得 API 序列,并以此作为特征.由于每一个 API 序列便包含了调用顺序信息,这便在一定程度上弥补了部分工作没有调用顺序信息的缺憾.Ahmed 等人[33]也是通过动态分析的方式,在运行时提取特征.他们分别使用空间特征和时序特征,空间特征由 API 的参数、返回值构成;时序特征由 API 调用序列构成.Park 等人[34]没有直接使用 API 序列来作为特征,而是通过在沙箱中动态运行被分析程序,将获得的系统调用序列作为特征.Eskandari 等人[35]是通过静态分析的方式提取 API 和 CFG(控制流图).他们并没有单独将待测程序的 API 作为特征,而是把 API 的信息附着到 CFG 相应的调用边上,组成一个特殊的"API-CFG"结构,进而作为特征.

1.4 小 结

本节主要介绍了恶意软件智能检测的常用特征.总的来看,大部分特征需要人工提前制定,但近些年来,也有相关工作开始尝试利用神经网络自动提取特征.在获取特征时,主要分为静态获取与动态获取 2 种.静态获取的特征侧重于从文件数据本身去发掘待测程序的特性;而动态获取的特征则侧重于发掘程序运行时的行为特性.

静、动态获取程序特征都有自身的优势或局限性.在众多的特征中,程序的 API 及系统调用信息、运行时状态信息最能反应程序的行为,相对而言最为重要;对于字节序列、操作码序列、神经网络自提取特征都难以直接反应程序的行为,重要性次之;而可阅读字符串、DLL 相关信息等都常用于估计待测程序行为,重要性相对较低;对于文件头部信息和熵等特征,则更加难以反应程序的行为信息,重要性最低.

由于各种特征各有利弊,在实际操作过程中,智能检测相关工作通常会同时用到多种特征.如 Schultz 等人[4]同时使用了字节序列与动态链接库相关信息来作为特征;Li 等人[3]则运用了待测程序的文件头、DLL 等特征;Masud 等人[8]则结合了字节序列、操作码序列及动态链接库 3 种特征;Kumar 等人[13]同时将文件头部信息与熵作为特征等.

综上所述,我们将常用特征的获取方式、用到这些特征的代表性相关工作及特征优缺点进行了总结,如表 1 所示:

Table 1 SummaryTable of Common Features

表 1 常用特征总结表

特征	特征获取方式	相关工作	特征优点	特征缺点
字节序列	静态获取	文献[4-5,7-10,17,36-37]	1) 容易获取; 2) 反映了程序最直观的特征.	1) 特征过于底层,可能还需要进一步提炼高层特征; 2) 字节序列的切分方式需要多次尝试、调优,工作量较大.
操作码序列	静态或动态获取	文献[8,17,26,28-29,38]	包含丰富的语义信息.	1) 遇到代码混淆或反动态分析等情况时,获取该特征存在难度; 2) 动态分析时开销较大.
可阅读字符串	静态获取	文献[4,11]	特征直观且容易获取.	通常需要人工筛选.
文件头部信息	静态获取	文献[3,7,12-14,18,39]	1) 容易获取; 2) 包含大量的程序宏观信息; 3) 可获取的特征种类丰富,方法多样.	获取后的特征数量较大,需要择优保留.
熵	静态获取	文献[3,13-16,39]	1) 容易获取; 2) 有一定的理论及统计经验作为基础.	提取特征方式较为单一,容易被攻击者针对.
动态链接库(DLL)相关信息	静态获取	文献[3-4,8,12,17-18,39]	1) 容易获取; 2) 包含丰富的语义信息.	提取特征方式较为单一,容易被攻击者针对.
API 及系统调用	静态或动态获取	文献[30-35,40-42]	1) 包含丰富的语义信息; 2) 能够直观地反应程序行为.	1) 遇到代码混淆或反动态分析等情况时,获取该特征存在难度; 2) 动态分析时开销较大.
运行时状态信息	动态获取	文献[19,22-23,25,43]	1) 包含丰富的语义信息; 2) 能够直观地反应程序状态信息.	1) 遇到反动态分析等情况时,获取该特征存在难度; 2) 动态分析时开销较大.
神经网络自动提取特征	静态获取	文献[10,19-20,44-45]	1) 减少人工参与; 2) 相比于人工制定特征,自动提取的特征可能更有效.	1) 需要设计合理的网络结构; 2) 需要大量调参; 3) 需要大量数据; 4) 程序语义过于复杂,自动化提取特征存在客观难度.

2 特征处理

在确定常用特征之后,需要对提取的特征进行处理.在恶意软件智能检测中,特征处理主要包含挑选重要特征、去除冗余特征、对特征降维等.其对于智能检测的主要作用是提升训练效率、帮助分类器收敛,即尽可能地帮助分类器达到一个最佳的效果,进而使得智能检测足够准确、可靠.本节主要介绍恶意软件智能检测相关工作中的常用特征处理方法.

2.1 信息增益

信息增益可用于衡量相应变量能够增加信息量的程度.在恶意软件智能检测中,信息增益可以用来衡量相应特征对于分类恶意软件的重要程度,进而作为特征筛选的依据.信息增益的一般使用方式是:在给定某些场景下的熵计算方式后,如果一个特征越能够降低熵值,则说明该特征越重要[17],进而使用者就可以按照特征的重要程度来选取特征.

在恶意软件智能检测中,信息增益已经广泛用于特征选择[5,8-9,17-18,33,46-47].如 Masud 等人[17]就是运用信息增益的方法分别挑选出最重要的 500 个待测程序的字节特征、动态链接库特征等.Zhang 等人[47]则利用信息增益来确定有效的系统调用特征.相关工作使用信息增益的方式大同小异,在此不再赘述.

2.2 去除冗余特征

在特征提取阶段完成后,得到的特征集合中可能存在一些冗余特征.冗余特征是指那些不能提供任何有效信息的特征,它们无法为训练分类器正确识别恶意软件提供帮助[2].对此,需要对这些冗余特征进行去除[2,12,23,46].

Shafiq 等人[12]认为,对于不同的待测程序而言,冗余特征对应的特征值不会发生变化或发生变化的幅度过大(变化幅度过大可能意味着该特征值是完全随机的).他们会直接将这些特征值稳定不变或变化值超过预先设定阈值的特征进行移除.Ye 等人[2]则提出可以针对不同的应用场景为提取的特征设计一个打分标准,然后在此基础上对不同的特征进行打分,并择优保留排名靠前的特征.Ghiasi 等人[23]则维护了一个特征频率表,对出现频率较低的特征进行去除.

2.3 主成分分析

主成分分析(principal components analysis,

PCA)是一种统计分析、简化数据集的方法,它常用于降维.它的本质是利用正交变换对一系列可能相关的变量进行线性变换,进而转换为1组不相关的变量.该方法针对高维数据降维非常有效.对于待分析的高维数据,如果其特征之间高度相关,或者其包含很多冗余特征,则PCA方法能够对其起到很好的降维作用.

Siddiqui等人[48]使用PCA技术将初始特征变量数从877降到了最终的146,降维效果很明显.Zhang等人[47]和Shafiq等人[12]同样使用了PCA来对高度相关的特征进行降维.但是特征降维可能会导致被降维数据信息丢失(比如降维后数据方差减少).所以在进行降维时,需要综合考虑降维程度与信息丢失之间的平衡[12].

2.4 小波变换

小波变换是一种常用的变换手段,相较于傅里叶变换而言,小波变换能够更好地从信号中提取信息.在恶意软件的智能检测中,小波变换也可以用于特征降维,或转换特征表示形式进而得到新特征[12,15-16,49].降维的本质在于将原始特征表示为1组小波基的线性组合,然后忽略其中的部分非重要项,即可达到降维目的.而转换特征表示形式并提取新特征则依靠小波变换从原始特征中提取多个定长项,并依据每一项的能谱来作为特征.

Shafiq等人[12]在处理特征时,使用的是哈尔小波变换.通过小波变换,能够使用先前层级的系数推出更高层级的特征.这些高层级的特征代表的往往是噪声或细节信息,属于非重要项.所以在表示时,可以适当的移除某些高层级项,这样便达到了降维的效果.除了特征降维,小波变换的另一个用途是将特征拆分为多个固定长度的项,而每一个项的能谱可以单独作为特征.Wojnowicz等人[15]便针对熵特征使用了小波变换,得到了多个不同层级的定长项.他们使用这些不同项的能谱来作为特征,并用于训练分类器.

2.5 运用分类器处理特征

在恶意软件智能检测中,分类器不仅可以用于对待测软件进行检测,还可以作为特征处理的工具.在处理特征时,常用的分类器有随机森林、自组织映射神经网络等[22,48,50].

分类器可以帮助筛选特征.例如,在面对特征集合时,可以使用控制变量法,通过逐个移除某一特征后再重新训练分类器,并观察分类器准确率或基尼系数的下降程度来判断相应特征的重要性.Siddiqui等人[48]使用该方法在随机森林模型上进行实验,并将分类器准确率或平均基尼系数下降至小于10%的相应特征进行移除.通过筛选,能够将原始的特征变量数量从877降低至84.为了保证实验结果的可靠程度,还可以同时使用多个分类器进行实验观测.Raman[51]同时使用了多个算法生成的分类器,如:IBk,J48,Ridor,随机森林等.在多个分类器下,他们通过特定的算法来观测单一特征对这些分类器识别准确率的影响,并最终挑选出了7个能够对恶意软件识别起到关键作用的特征.

分类器还可以直接用于降维.例如,使用自组织神经网络来降低特征维度.Burnap等人[22]将提取的特征数据输入到自组织神经网络中,不仅得到了降维后的特征,还能够通过该网络获得特征数据的可视化表示,便于研究者进行分析.

2.6 嵌 入

embedding是一种将离散变量转换为连续向量表示的常用方法.embedding能够减少离散变量的空间维数,embedding后的结果还能体现不同实体间的关系.在自然语言处理、搜索排序、推荐系统等相关领域,embedding已经被广泛使用.

在神经网络自动提取程序特征的智能检测工作中,也会用到embedding.由于直接利用程序字节值会存在许多问题[10],如无法根据字节值来衡量字节之间的语义相似性.对此,相关工作可能需要使用embedding的方法来对程序的字节进行转换.这样不仅降低了输入数据的维度,而且转换后的数据还可以体现不同实体间的语义关系.

Raff等人[10]就是使用embedding的方法来改变程序字节表示形式.他们在使用的CNN中加入了embedding层,达到了无监督学习训练该层的目的,其具体结构如图5所示.且当CNN训练好后,相应的embedding层也会训练完成.在检测过程中,当程

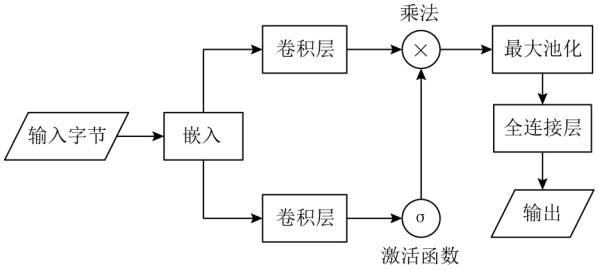

Fig. 5 CNN structure diagram using embedding layer

图5 使用embedding层的CNN结构图

序字节输入到 CNN 中时,由于 embedding 层的存在可以直接对其进行转换,转换后的结果会直接用于检测.

2.7 小 结

本节主要介绍了恶意软件智能检测的常用特征处理方法.特征处理主要关注的是如何精简已有的特征.从它们的功能来看,无论是挑选重要特征、去除冗余特征还是特征降维,本质上都是对特征的精简,旨在去除特征中无用或用处不大的部分,进而使得分类器能够聚焦于有用的特征.此外,随着深度学习的发展,对于一些不需要直接由人工制定特征的场景,特征处理面对的对象就成为了最基本的程序数据.此时特征处理关注的应该是如何将程序数据以合理的形式呈现给分类器,如使用嵌入的方法,将程序数据转换到新的表征空间等.而这是为了便于分类器更好地自动提取程序特征,进而促进智能检测效果.综上,我们将特征处理方法进行了总结,如表 2 所示:

Table 2 Summary of Feature Processing Methods

表 2 特征处理方法总结

特征处理方法	特征处理优点	特征处理缺点
信息增益	能够有效地用于筛选特征,使用较为广泛.	所需计算量较大.
去除冗余特征	能够去除作用不明显或明显存在问题的特征.	1) 方法较为简单直观,需要根据经验设置阈值; 2) 能够筛选的特征较为有限,无法从深层次上精简、提炼特征.
主成分分析	1) 降维后各主成分之间正交,可消除原始成分间的相互影响; 2) 计算简单,能够在一定程度上降低人工参与.	1) 容易导致特征信息丢失,需要在特征降维与信息丢失间寻找平衡; 2) 属于无监督学习,降维效果有局限性.
小波变换	能够较为有效地降维.	1) 相对使用较少; 2) 同样存在特征信息丢失的问题,需要在特征降维与信息丢失间寻找平衡.
运用分类器处理特征	1) 能够降低人工参与; 2) 实现简单,原理直观.	1) 需要消耗算力训练分类器; 2) 特征筛选的准确性取决于分类器的效果.
embedding	1) 能够使特征的表征体现实体间的语义关系; 2) 同时能够起到降维作用.	需要训练好 embedding 层,表征的结果取决于 embedding 的准确性.

3 分类器

在获得处理完的特征后,便会将其输入到分类器中,以获取检测结果.在恶意软件智能检测相关工作中,分类器主要分为机器学习分类器与深度学习分类器.尽管本节是将多种分类器分开介绍,但对于智能检测而言,现有的工作通常会包含多种分类器,从而进行分类器之间的对比,进而获取最优的实验效果;或结合不同分类器,达到更好的效果.

3.1 机器学习分类器

3.1.1 支持向量机

支持向量机(support vector machine, SVM)是一种适用于分类问题的监督学习模型,常用于二分类.它能够很好地应对线性与非线性分类问题.在线性分类问题中,SVM 能够在数据间形成一个超平面,进而将数据分类.在非线性分类问题中,SVM 需要结合核技巧,将特征空间转换为高维空间,然后再构建超平面进行分类[8].

在智能检测领域,SVM 常用于二分类场景,即判断待测软件是否为恶意.在当前的相关工作中,SVM 已经被广泛使用[8,13,17,25,30,33].如 Li 等人[3]将待测软件的文件头、DLL 等特征作为 SVM 的输入来进行分类.Kolter 等人[5]则将获得的字节序列特征送入 SVM.而 Masud 等人[8]则综合考虑了字节序列、操作码、DLL 等多种特征来作为 SVM 的分类依据.

3.1.2 决策树与随机森林

决策树是一种常用的监督学习模型,决策树分类算法是一种基于实例的归纳学习方法,它能从给定的无序的训练样本中,提炼出树型的分类模型.而随机森林则是由多颗决策树组成,它会以投票的形式,将所有决策树的结果众数作为自身的输出结果.决策树与随机森林也常用于恶意软件智能检测.

Kolter 等人[5]在其工作中使用了多种分类器,其中就包含决策树、以及决策树的变体“Boosted Tree”.Elovici 等人[6]则考虑了决策树变体“J48”.Eskandari 等人[32]则在使用决策树的同时,还使用了随机森林.Raman[51]使用的分类器种类更多,除了随机森林外,他们还使用了多种决策树变体,如:“IBk,J48,J48 Graft,Part”等.

3.1.3 朴素贝叶斯与贝叶斯网络

朴素贝叶斯分类器也是一种有监督分类模型，它的本质是一个条件概率模型.它是在假设特征之间相互独立的条件下，运用贝叶斯定理来建立的概率分类器.贝叶斯网络则是一种有向无环的概率图模型，主要用于表示变量之间的依赖关系.图中用有向边连接的节点是非条件独立的，二者之间具有条件概率值；而无边相连的节点是相互条件独立的.朴素贝叶斯与贝叶斯网络也常作为智能检测的分类器.

在恶意软件智能检测的相关工作中，Ahmed 等人[33]使用了朴素贝叶斯分类器来进行恶意软件识别.除了使用朴素贝叶斯分类器外，Kolter 等人[5]还运用了朴素贝叶斯的变体“Boosted Bayes”分类器来进行分类.Schultz 等人[4]则同时使用了多个朴素贝叶斯分类器，并将多个分类器的投票结果的众数来作为最终结果.而在使用朴素贝叶斯分类器的同时，Masud 等人[8]还使用了贝叶斯网络分类器.

3.1.4 K 近邻算法

K 近邻算法(K-nearest neighbor, KNN)算法是根据不同特征值之间的距离来进行分类的机器学习方法.给定一个输入数据，KNN 会统计与该输入最近的 K 个点所属的类别，并以投票的方式，将数量最大的那一类作为该输入的预测结果.KNN 被提出的时间较早，属于较为简单的机器学习算法，而使用 KNN 的智能检测相关工作也偏向于早期.

Firdausi 等人[52]将动态获取的程序行为信息作为特征，并交由 KNN 进行检测.Abou-Assaleh 等人[37]则是将 n-gram 得到的字节序列特征作为 KNN 的输入，进而进行分类.

3.2 深度学习分类器

3.2.1 人工神经网络

人工神经网络(artificial neural network, ANN)[53]是一种模仿生物神经系统的网络结构.它主要由一批相互连接的“神经元”构成.相连的神经元之间具有相应的权重值.在网络的训练过程中，会通过反向传播算法，不断地调整各权重值来使得网络模型满足任务的需求.

一些早期的智能检测工作会考虑采用 ANN.如 Moskovitch 等人[38]是将提取的操作码序列特征作为 ANN 的输入，进而进行检测.他们[36]在另一份工作中则是用 n-gram 提取字节序列，再结合 ANN 进行检测.而 Elovici 等人[6]同时将 n-gram 提取的字节序列与文件头部信息特征作为 ANN 的输入.由于 ANN 过于简单，近些年基于深度学习技术的智能检测工作已经不再使用 ANN.相关工作逐渐开始尝试 CNN，RNN 等其他功能性更强的神经网络.

3.2.2 卷积神经网络

卷积神经网络(convolutional neural network, CNN)是 2012 年前后兴起于计算机视觉领域[54]的一种典型神经网络模型，通常包括卷积层、池化层和全连接层等.CNN 的卷积与池化层使得它能够很好地提取数据的 2 维特征，因此它能够在图像识别相关领域表现优异.近年来，相关工作也开始将 CNN 用于恶意代码智能检测.

在 CNN 与智能检测相结合的相关工作中，部分工作先将待测程序转换为图片的形式[19-20,45]后，再利用 CNN 进行检测.如 Yan 等人[20]先以单个字节为最小单元，再按照此最小单元将待测程序转换为灰度图，随后再用 CNN 进行检测.而 He 等人[45]则是将程序转换为 RGB 图.他们以 3B 为一个单位，使每 3B 分别对应 R，G，B 这 3 个信道.在这些转化为图的工作中，本质上是将程序的相似性与图的相似性关联起来.

对于将程序转化为图的相关工作，会让程序字节值与像素值相对应，即直接利用程序的字节值.而这些直接利用字节值的方法不被 Raff 等人[10]认可.他们认为，对于神经网络而言，如果直接利用字节的值，则字节值越相近意味着字节语义越相近.而这明显是错误的，因为值相近的字节并不意味着对应的语义也相近.对此，他们先用 embedding 的方法将程序字节转化为向量，随后再利用 CNN 进行检测.在 Raff 等人的工作中，在采用最大池化策略的同时，还设置了合适大小的滑动窗口，旨在捕捉到处于不同位置的重要程序特征.而 Tobiyama 等人[43]并没有直接将待测程序的静态特征作为 CNN 的输入.通过动态运行待测程序，他们获得了程序行为对应的 API 序列，然后用循环神经网络(recurrent neural network, RNN)提取 API 序列的特征，随后再将 RNN 的特征提取结果作为 CNN 的输入，进而进行检测.Chai 等人[41]则是先通过 CNN 提取 API 序列的局部特征，再将提取的结果作为图卷积神经网络的输入进而得到全局特征.

3.2.3 循环神经网络

循环神经网络是一类以序列数据为输入，在序列的演进方向进行递归，且所有节点(循环单元)按链式连接的递归神经网络[55].循环神经网络具有记忆性、参数共享等特性，在对序列的非线性特征进行学习时具有一定的优势，目前已广泛应用于自然语言

处理领域.由于程序代码是由计算机语言编写的,跟自然语言有类似之处,因此在有些智能检测相关工作中,也使用了 RNN.

Shukla 等人[44]使用 RNN 来检测恶意软件,并与基于 CNN 和基于隐马尔可夫模型(HMM)的方法进行了对比,发现在其检测场景下,使用 RNN 的效果更好.Tobiyama 等人[43]则将 RNN 与 CNN 结合使用,他们用 RNN 提取的特征来作为 CNN 的输入,进而实现检测.而 Mathew 等人[40]则使用 RNN 的变体 LSTM(长短记忆网络)来作为分类器,旨在对提取的 API 序列特征进行检测.

3.2.4 图卷积神经网络

图卷积神经网络(graph convolutional network,GCN)是一种能对图数据进行深度学习的方法[56].GCN 可以直接作用于图数据,它能在图数据上进行卷积操作.就目前而言,使用 GCN 的恶意软件智能检测相关工作较少,仍处于早期阶段.

在智能检测领域中,GCN 主要与 API 序列特征结合使用.Oliveira 等人[42]借助了图卷积神经网络来进行检测.他们先获取了待测程序的 API 序列,并分析出 API 的调用顺序,随后据此构建 API 之间的行为图.最终他们将行为图作为图卷积神经网络的输入,进而实现检测.

3.3 小 结

本节主要介绍了恶意软件智能检测的常用分类器.在早期的相关工作中,智能检测偏向于使用传统的机器学习分类器.而近些年来,越来越多的深度学习分类器被引入到智能检测中.传统机器学习分类器是以人工提取的特征作为输入;而深度学习分类器既能以人工提取的特征为输入,也可自动提取程序特征.此外,一些分类器只在较早的工作中出现,如 KNN 和 ANN.另一方面,一些新式的深度学习分类器则在近些年才得以应用,且应用的较少,如 GCN.未来可能会有更多的深度学习分类器被引入到智能检测中.最后,我们对使用过这些常用分类器的代表性工作进行了总结,如表 3 所示:

Table 3 Summary Table of Common Classifiers

表 3 常用分类器总结表

分类器	相关工作	分类器优点	分类器缺点
支持向量机(SVM)	文献[3,5,8,11,17,32-33,35-36,50,52]	1) 在样本数量较少时,可发挥出色性能; 2) 可用于解决非线性问题.	1) 难以应用于大规模数据场景; 2) 需要人工制定特征.
决策树与随机森林	文献[5-8,12-14,18,29,32-33,35-36,38,50-52]	计算开销相对较小且容易并行化.	1) 需要人工制定特征 2) 容易过拟合; 3) 难以应用于大规模数据场景.
朴素贝叶斯与贝叶斯网络	文献[4-5,7-8,11-14,32-33,35-36,38,52]	1) 在小规模数据上表现较好; 2) 算法简单直观,易于实现.	1) 难以应对大规模数数据,近年来使用相对较少; 2) 需要人工制定特征.
K 近邻算法(KNN)	文献[7,13,37,52]	1) 在小规模数据上表现较好; 2) 算法简单直观,易于实现.	1) 计算复杂性高,难以应用于大规模数据场景,近年来使用相对较少; 2) 需要人工制定特征.
人工神经网络(ANN)	文献[6,36,38-39,50,52]	可自动提取程序特征.	ANN 结构简单,难以应对复杂的任务场景,近年来使用相对较少.
卷积神经网络(CNN)	文献[10,19-20,41,43-45]	1) 可自动提取程序特征; 2) 能够捕捉待测程序的结构信息.	CNN 在智能检测中的使用属于早期阶段,尚不成熟,难以学习到程序的本质.
循环神经网络(RNN)	文献[3,40,43-44]	1) 可自动提取程序特征; 2) 能够捕捉待测程序的上下文信息.	RNN 在智能检测中的使用属于早期阶段,尚不成熟,难以学习到程序的本质.
图卷积神经网络(GCN)	文献[41-42]	1) 可自动提取程序特征; 2) 能够捕捉待测程序的上下文信息与结构信息.	1) GCN 在智能检测中的使用属于早期阶段,尚不成熟; 2) 需要将输入处理为图结构的形式.

4 智能检测面临的问题

随着恶意软件智能检测逐渐被广泛应用,其面临的问题也不容忽视.本节主要介绍当前智能检测相关工作中所存在的主要问题,包括数据收集中的问题,静、动态检测时的局限性,以及对抗样本威胁.

4.1 数据收集相关问题

4.1.1 标准数据集现状

在 Windows 平台下,就现有的标准数据集而言,目前只有“EMBER”[57]与“BIG 2015”[58].出于对版权问题的考虑,EMBER 数据集中只包含了良性软件与恶意软件的特征,总数为 110 万个.而 BIG 2015 则包含了恶意软件,且为了避免造成危害,均被

去除了文件头部.总的来看,现有的开源标准数据集较少,无法满足学术界与工业界的需求.对于智能检测相关工作而言,普遍还是需要自行收集数据.

4.1.2 收集恶意软件相关问题

在恶意软件智能检测中,相关工作收集恶意软件的主要方法是从一些网站上[59-60]直接下载,或部署蜜罐进行收集.然而,恶意软件的收集很容易存在数据不平衡的问题,进而影响分类器的训练效果.

以蜜罐收集恶意软件为例.蜜罐本身只能收集它能够收集的数据[49],进而收集的数据存在不平衡性.具体而言,某些恶意软件需要在特定的环境下才会展现恶意行为,比如需要与特定的应用程序进行交互后才能展现恶意行为.在这种情况下,蜜罐必须具备相应的环境才能捕捉这类恶意软件,否则便无法收集.显然,这会导致很多不满足蜜罐环境的恶意软件被遗漏.此外,有些恶意软件还会检测出蜜罐的存在,进而避免被收集.因此,通过蜜罐获取的恶意软件种类只是众多恶意软件种类中的一小部分,这样收集下来的数据集必然存在数据不平衡的问题.

4.1.3 收集良性软件相关问题

在训练分类器时,除了恶意软件外,还需要有充足的良性软件.相比于收集恶意软件,收集良性程序的难度可能更高[49].对于良性软件的收集,大部分工作是在 Windows 系统中直接获取系统程序或下载一些热门的应用程序.然而这样的收集方式可能会导致分类器泛化能力不足,影响分类器可用性.

首先,Windows 系统程序本身可能带有一些固定的字符串,如“Microsoft Windows”.当智能检测分类器使用这些数据进行训练时,可能会将这些固定字符串作为区分恶意软件与良性软件的重要特征,而无法学习到它们之间的本质区别,最终导致分类器泛化能力不足.

其次,Windows 系统程序多半是不加壳的;而为了躲避检测,大量的恶意软件都会加壳.在这种情况下,训练集会出现明显的不平衡:训练集中的加壳程序往往是恶意的.而这种训练集的不平衡性会极大地影响分类器的泛化能力.具体而言,分类器在进行训练时,会将壳的存在与否作为判别恶意软件的重要依据[61].而在真实场景下,很多良性软件也会加壳(比如通过加壳来防止被破解).这样一来,先前的分类器可能会盲目地将这类带壳的良性软件误判为恶意软件,进而极大地影响分类器的可用性.

4.2 打标签相关问题

对于收集的程序数据集,还需要对它们打标签.现有的打标签方式主要依赖于杀毒软件或恶意软件检测平台等.如将程序上传到 VirusTotal(VirusTotal 上集成了多个杀毒软件的在线检测服务)进行打标签.在打标签时,相关工作会制定自己的标准来决定程序的恶意性.如以投票的形式[62-65],根据预先设定的阈值,当超过阈值的杀毒软件将待测程序识别为恶意软件时,则为其打上恶意的标签;反之则打上良性标签.此外,还可以预先认定一批权威的杀毒软件[66],当这些权威的杀毒软件同时将目标对象识别为恶意时,则打上恶意标签,反之亦然.

然而,这些打标签的正确与否本身依赖于杀毒软件自身的可靠性.而在现实场景下,杀毒软件自身往往会存在问题.首先,杀毒软件会不断进行更新,其决策边界会发生变化,先前制定好的投票阈值不一定能够满足更新后的需求.其次,对于先前未见过的待测程序,杀毒软件的泛化能力可能不足,它们无法准确的检测待测程序,进而它们所打的标签也是不可靠的.再者,对于一些投票结果处在阈值边缘的待测程序,对它们所打的标签可能存在误报或者漏报.

4.3 静态检测局限性

在恶意软件智能检测中,静态检测需要依赖于静态特征的提取.然而,待测程序常常会使用加壳技术来对抗静态特征提取.对于加壳的待测程序而言,很难获得其反汇编代码.而没有反汇编代码则意味着无法提取与反汇编代码相关的特征.例如,由于没有反汇编代码,静态获取操作码相关特征将变得不可行.就目前而言,在静态检测领域,还没有较好的办法能够针对加壳待测程序提取对应的反汇编代码.因此,在面对加壳程序时,静态智能检测往往需要从其他的角度提取特征[61],如使用熵特征等.

此外,即使待测程序未加壳,现有的反汇编技术也不能保证反汇编结果完全准确.例如,可能会出现静态识别间接跳转不准确,导致反汇编结果出错的情况.这也可能在一定程度上给相关特征的提取带来麻烦.

4.4 动态检测局限性

对于动态检测而言,它会在待测程序运行时提取动态特征.然而,动态提取特征的过程同样会面临很多问题.

首先,现阶段的动态检测方法通常是将待测程序运行在虚拟环境中,并动态监测待测软件行为,进而提取特征.但是,有些恶意软件能够检测到虚拟环境的存在,并且在虚拟环境中隐藏恶意行为[67-69].这样一来,其相关特征便不能被很好地提取.为了应对

这一问题,可以通过强制执行某些程序路径的方式来进行弥补,迫使程序执行某些隐藏的路径,进而表现出恶意功能[70-71].然而,这种探索路径的方式存在局限性,通常不能将待测程序的所有路径遍历完,且很容易遭遇路径爆炸等问题.

其次,恶意软件可能需要在特定的环境下才能执行相应恶意功能.比如,某些恶意软件对当前的操作系统版本有要求,或对是否安装了特定的软件有要求[72]等.这对运行待测程序的虚拟环境要求很高,且很难考虑周全.

另外,在进行动态检测时,除了需要消耗大量的计算资源,还可能面临一些安全上的问题.如一些恶意软件执行时,需要一些内核级的特殊权限、网络连接等,这都可能会带来一些安全上的隐患[49,73].现有的一些工作提出了一些缓解措施,如在隔绝网络的条件下,可以使用网络模拟器来给恶意软件提供一些虚假的网络回复[74-75]等,进而在不影响恶意软件行为的前提下,避免让恶意软件直接与真实的网络环境相接触.

4.5 对抗样本威胁

对抗样本最早是由 Szegedy 等人[76]提出的,对于一个目标模型能够正确识别的原始输入,通过添加一些肉眼不可见的细微改动,使得模型发生误判,则改动后得到的输入就是对抗样本.对抗样本示意图如图 6 所示.其中,"噪声"即为人为构造的扰动.当扰动添加到原始图片上之后,便可得到对抗样本.

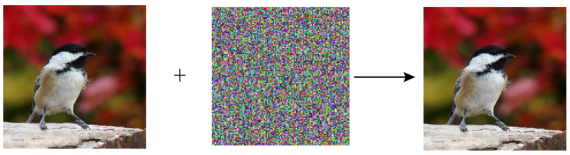

Fig. 6 Adversarial example

图 6 对抗样本

对抗样本的兴起让人们开始逐渐关注起人工智能的安全性问题.目前而言,对抗样本已经广泛存在于图像识别、语音识别、目标检测等多个领域.而在恶意软件的智能检测中,其所使用的人工智能模型同样存在被对抗样本攻击的风险.与其他领域不同的是,在恶意软件对抗样本生成过程中,只要不影响恶意软件正常功能,攻击者可以对恶意软件做任意的改动.

针对恶意软件智能检测的对抗样本相关工作主要分为 2 类,即白盒攻击和黑盒攻击.白盒攻击是指攻击者知道智能检测目标模型的所有信息,随后在此基础上对目标模型进行攻击.而黑盒攻击则意味着攻击者不知道模型内部的任何信息.相比于白盒攻击,黑盒攻击场景更加符合实际.

对于白盒攻击而言,攻击者可以直接依据目标模型来计算目标程序可修改位置的梯度信息,并依据,梯度信息来改动相应位置字节,进而构造对抗样本.上述攻击流程基本上是当前白盒工作的主要做法.对于白盒工作而言,它们之间的主要区别在于如何确定程序可修改位置.大部分工作在制定修改位置时,都会考虑不破坏恶意软件的功能.如 Kolosnjaji 等人[77]先在文件末尾添加字节后,再进行修改.Kreuk 等人[78]则是选择在新增的节里进行修改.而 Demetrio 等人[79]认为,只需修改程序头部信息即可.然而,Liu 等人[80]在进行修改时,并没有考虑不破坏恶意软件的功能,他们直接对一整个程序的所有字节进行修改.这种方式得到的对抗样本往往是无法正常运行的,更无法保护恶意软件的功能.

对于黑盒攻击而言,攻击者不知道模型的内部信息.相比于白盒攻击,黑盒攻击的要求更为严格,且实用性更强.在黑盒攻击中,部分场景要求较为宽松,允许知道模型输出的置信度;而在严格的场景下,攻击者无法获取置信度,而只知道判别结果.甚至在有些情况下,黑盒攻击还会被限制询问次数.

尽管黑盒攻击要求更严格,但也意味着其实用性更强,且方法更为多样.而在黑盒场景下,同样需要遵循不破坏恶意软件功能的原则.因此,在生成对抗样本的过程中,针对目标程序的改动操作种类比较有限[81-82],常用的合法改动操作如表 4 所示.而相关工作会在有限的操作集合里进行方法上的创新.

Table 4 Common Actions of Black-box Attacks

表 4 黑盒攻击常用操作

序号	操作集合
1	在二进制程序尾部添加字节
2	在节的未使用处添加字节
3	创建新的节
4	修改已有的节名
5	删去签名信息
6	修改二进制程序的调试信息
7	修改头部校验和
8	对指令进行等价替换
9	在导入表中添加未被使用的函数
10	创建新的程序入口点,并重新跳回
11	压缩或解压程序文件

如 Anderson 等人[82]将强化学习运用到了对抗样本生成中,而强化学习的操作空间包括了一些不会改变恶意软件功能的操作,如创建新的节、在程序末尾添加字节、程序加壳或脱壳等.Song 等人[66]则设计了一套启发式算法,在这些允许的改动操作基础上,通过不断迭代,来尝试添加各种改动,进而构造对抗样本.

还有一些方法则不考虑在原始恶意软件上构造对抗样本,而是直接在提取出的特征上构造对抗样本.此类工作通常会提前假设已知目标模型使用的特征类别,随后再针对这些特征来进行改动[83-85]等.但是这样生成的对抗样本不是实际的可运行程序,也无法还原到对应的程序.具体而言,Rosenberg 等人[83]就是直接针对 API 序列构造对抗样本.他们提前假设已知目标模型是以 API 序列来作为特征,进而直接针对 API 序列进行改动.在他们的工作中,会先根据目标模型训练出一个替代模型,随后在替代模型上运用梯度下降的方式修改 API 序列,进而得到对抗样本,随后再结合迁移攻击来威胁目标模型.然而,由于改动的对象是 API 序列,所以相应的对抗样本也是 API 序列,且无法还原到程序上.即这样的对抗样本不是真正的可运行程序.Hu 等人[84]则利用生成对抗网络(generative adversarial network, GAN)来构造对抗样本,他们操作的对象是特征向量.其中,GAN 的判别器用于模拟目标网络,而生成器则用于产生改动,进而构造对抗样本.然而这种方法获得的对抗样本同样不是可运行程序.

对于黑盒工作而言,相关工作仍处于早期阶段.对于那些无法正常运行的对抗样本,难以在真实场景下对智能检测构成威胁.而对于可运行的对抗样本,则需要引起我们的重视.在此,我们将一些代表性的黑盒相关工作总结在表 5 中,表 5 中主要包含各黑盒工作的攻击场景、攻击方式、操作对象及是否影响原始恶意软件功能.

Table 5 Summary of Black-box Attacks

表 5 黑盒攻击总结

相关工作	攻击场景	攻击方式	操作对象	是否影响原始功能
文献[84]	黑盒(不需要目标模型置信度信息)	GAN	特征向量	是
文献[85]	黑盒(有、无目标模型置信度信息均可)	GAN 与启发式算法	API 序列	是
文献[83]	黑盒(不需要目标模型置信度信息)	迁移攻击	API 序列	是
文献[82]	黑盒(不需要目标模型置信度信息)	强化学习	二进制程序	否
文献[66]	黑盒(不需要目标模型置信度信息)	启发式算法	二进制程序	否
文献[86]	黑盒(需要目标模型置信度信息)	启发式算法	二进制程序	是

4.6 小 结

本节主要介绍了当前恶意软件智能检测所面临的主要问题.针对这些问题,一些缓解措施已经被提出,但仍无法彻底解决.不过这些问题的存在,也意味着相关方向可能会成为未来研究的关键与重点.

1) 就数据集收集而言,目前标准的数据集较少,且均为非完整程序.而缺乏标准数据集则会对分类器的效果评估带来明显影响.此外,现在也没有专门用于对抗训练的数据集,不利于提升分类器的鲁棒性.

2) 就数据集打标签而言,打标签的准确性依赖于打标签软件的准确性、评判阈值选择等问题.在面对难以判断的情况时,依旧需要专家经验来辅助分析.

3) 就静态检测而言,代码混淆相关技术依旧是问题的关键.对此,可以考虑使用动态分析辅助静态分析的方式来进行缓解,如先动态分析进行脱壳,随后再进行静态分析.

4) 就动态检测而言,也会面临诸多局限性,如恶意软件对抗动态分析、软件运行有特定环境要求等.对此,可以考虑使用静态分析辅助动态分析的方式来缓解此问题,如强制执行某条分支路径等.

5) 就对抗样本威胁而言,这已经是人工智能模型中普遍存在的问题.而在智能检测领域,目前基本没有防御对抗样本的相关工作被提出.未来的工作可以考虑借鉴其他领域的防御方案来进行缓解,如对抗训练等.

5 总结与展望

本文总结了当前恶意软件智能检测相关工作的的研究概况.总的来看,将人工智能与恶意软件检测相结合已经成为了研究者广泛讨论的热点.在众多的研究工作中,研究的主要内容都离不开特征提取、特征处理、设计分类器这 3 个主要环节.因此,我们

从这 3 个角度对智能检测的相关工作进行了归纳与总结.另外,我们还总结了当前智能检测相关工作所面临的主要问题.根据我们对相关工作以及相关问题的总结与思考,在此,提出一些潜在的研究方向,具体为:

1) 使用人工智能相关技术进一步辅助特征提取.就目前而言,大部分智能检测工作仍倾向于人工制定特征,只有较少的工作能够直接应用深度学习技术来自动提取特征.受制于程序本身的复杂性,在未来相当的一段时间内,特征制定的过程应该依旧需要人工参与.但可以在此基础上,让人工智能进一步完善先前提取的特征.比如先通过人工制定一些低级特征,再利用深度学习来提炼出高级特征,随后再交由分类器进行检测.

2) 制定新的训练数据集.就智能检测训练所需的数据集而言,目前已有的公开标准数据集较少,且多半为非完整程序,去除了某部分或只包含提取的特征[57-58].对此,在未来的工作中,提出更多有价值的数据集也是很有必要的.

3) 考虑更多地应用图神经网络.早期的智能检测工作偏向于使用机器学习相关技术来进行智能检测,而后续的工作则逐渐开始引入深度学习相关技术.在现有的智能检测工作中,使用的深度学习模型主要以 CNN 和 RNN 为主,而最近兴起的图神经网络则被应用的相对较少.未来的研究工作可以考虑如何将图神经网络更好地应用于恶意软件智能检测中.

4) 提升模型防御对抗样本的能力.随着对抗样本在恶意软件智能检测领域的出现,如何有效防御对抗样本也应当被充分考虑.研究者可以根据现有的对抗攻击,有针对性地提出一些防御方案.此外,在方面可以充分借鉴其他领域的相关工作,如图像中的防御工作等.

总之,未来的恶意软件智能检测技术还有很大的发展空间,目前尚有大量的问题仍待解决.充分利用好人工智能相关技术的优势是未来的趋势.恶意软件智能检测还有很长的路要走.

参 考 文 献

[1] Gantz J F, Lee R, Florean A, et al. The link between pirated software and cybersecurity breaches, 247411 [R]. Singapore: The National University of Singapore, 2013

[2] Ye Yanfang, Li Tao, Adjeroh D, et al. A survey on malware detection using data mining techniques [J]. ACM Computing Surveys (CSUR), 2017, 50(3): DOI:10.1145/3073559

[3] Li Bo, Roundy K, Gates C, et al. Large-scale identification of malicious singleton files [C] //Proc of the 7th ACM on Conf on Data and Application Security and Privacy. New York: ACM, 2017: 227-238

[4] Schultz M G, Eskin E, Zadok F, et al. Data mining methods for detection of new malicious executables [C] //Proc of 2001 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2000: 38-49

[5] Kolter J Z, Maloof M A. Learning to detect and classify malicious executables in the wild [J]. Journal of Machine Learning Research, 2006, 7(99): 2721-2744

[6] Elovici Y, Shabtai A, Moskovitch R, et al. Applying machine learning techniques for detection of malicious code in network traffic [C] //Proc of Annual Conf on Artificial Intelligence. Berlin: Springer, 2007: 44-50

[7] Menahem E, Shabtai A, Rokach L, et al. Improving malware detection by applying multi-inducer ensemble [J]. Computational Statistics & Data Analysis, 2009, 53(4): 1483-1494

[8] Masud M M, Khan L, Thuraisingham B. A scalable multi-level feature extraction technique to detect malicious executables [J]. Information Systems Frontiers, 2008, 10(1): 33-45

[9] Raff E, Zak R, Cox R, et al. An investigation of byte *n*-gram features for malware classification [J]. Journal of Computer Virology and Hacking Techniques, 2018, 14(1): 1-20

[10] Raff E, Barker J, Sylvester J, et al. Malware detection by eating a whole exe [C] // Proc of the 32nd AAAI Conf on Artificial Intelligence Workshop. Menlo Park, CA: AAAI, 2018: 2-7

[11] Ye Yanfang, Chen Lifei, Wang Dingding, et al. SBMDS: An interpretable string based malware detection system using SVM ensemble with bagging [J]. Journal in Computer Virology, 2009, 5(4): 283-293

[12] Shafiq M Z, Tabish S M, Mirza F, et al. Pe-miner: Mining structural information to detect malicious executables in realtime [C] //Proc of the 12th Int Workshop on Recent Advances in Intrusion Detection. Berlin: Springer, 2009: 121-141

[13] Kumar A, Kuppusamy K S, Aghila G. A learning model to detect maliciousness of portable executable using integrated feature set [J]. Journal of King Saud University-Computer and Information Sciences, 2019, 31(2): 252-265

[14] Markel Z, Bilzor M. Building a machine learning classifier for malware detection [C] //Proc of the 2nd Workshop on Anti-malware Testing Research (WATeR). Piscataway, NJ: IEEE, 2014: DOI:10.1109/WATeR.2014.7015757

[15] Wojnowicz M, Chisholm G, Wolff M, et al. Wavelet decomposition of software entropy reveals symptoms of malicious code [J]. Journal of Innovation in Digital Ecosystems, 2016, 3(2): 130-140

[16] Gibert D, Mateu C, Planes J, et al. Classification of malware by using structural entropy on convolutional neural networks [C] //Proc of the 32nd AAAI Conf on Artificial Intelligence. Menlo Park, CA: AAAI, 2018: 2-7

[17] Masud M M, Khan L, Thuraisingham B. A hybrid model to detect malicious executables [C] //Proc of the 2007 IEEE Int Conf on Communications. Piscataway, NJ: IEEE, 2007: 1443-1448

[18] Bai Jinrong, Wang Junfeng, Zou Guozhong. A malware detection scheme based on mining format information [J]. The Scientific World Journal, 2014, 2014: No.260905

[19] Abdelsalam M, Krishnan R, HuangYufei, et al. Malware detection in cloud infrastructures using convolutional neural networks [C] //Proc of the 11th IEEE Int Conf on Cloud Computing (CLOUD). Piscataway, NJ: IEEE, 2018: 162-169

[20] Yan Jinpei, Qi Yong, Rao Qifan. Detecting malware with an ensemble method based on deep neural network [J]. Security and Communication Networks, 2018, 2018: No.7247095

[21] Nataraj L, Karthikeyan S, Jacob G, et al. Malware images: Visualization and automatic classification [C/OL] //Proc of the 8th Int Symp on Visualization for Cyber Security. New York: ACM, 2011 [2020-11-20]. https://doi.org/10.1145/2016904.2016908

[22] Burnap P, French R, Turner F, et al. Malware classification using self organising feature maps and machine activity data [J]. Computers & Security, 2018, 73: 399-410

[23] Ghiasi M, Sami A, Salehi Z. Dynamic VSA: A framework for malware detection based on register contents [J]. Engineering Applications of Artificial Intelligence, 2015, 44: 111-122

[24] Wang Lina, Tan Cheng, Yu Rongwei, Yin Zhengguang. Themalware detection based on data breach actions [J]. Journal of Computer Research and Development, 2017, 54(7): 1537-1548

[25] Mohaisen A, Alrawi O, Mohaisen M. Amal: High-fidelity, behavior-based automated malware analysis and classification [J].Computers & Security, 2015, 52: 251-266

[26] Ye Yanfang, Li Tao, Chen Yong, et al. Automatic malware categorization using cluster ensemble [C] //Proc of the 16th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM 2010: 95-104

[27] Khodamoradi P, Fazlali M, Mardukhi F, et al. Heuristic metamorphic malware detection based on statistics of assembly instructions using classification algorithms [C] // Proc of the 18th CSI Int Symp on Computer Architecture and Digital Systems (CADS). Piscataway, NJ: IEEE, 2015: No. 15701784

[28] Anderson B, Storlie C, Lane T. Improving malware classification: Bridging the static/dynamic gap [C] //Proc of the 5th ACM Workshop on Security and Artificial Intelligence. New York: ACM, 2012: 3-14

[29] Piyanuntcharatsr S S W, Adulkasem S, Chantrapornchai C. On the comparison of malware detection methods using data mining with two feature sets [J]. International Journal of Security and Its Applications, 2015, 9(3): 293-318

[30] Rieck K, Holz T, Willems C, et al. Learning and classification of malware behavior [C] //Proc of the 5th Int Conf on Detection of Intrusions and Malware, and Vulnerability Assessment. Berlin: Springer, 2008: 108-125

[31] Qiao Yong, Yang Yuexiang, Ji Lin, et al. Analyzing malware by abstracting the frequent itemsets in API call sequences [C] //Proc of the 12th IEEE Int Conf on Trust, Security and Privacy in Computing and Communications. Piscataway, NJ: IEEE, 2013: 265-270

[32] Eskandari M, Khorshidpur Z, Hashemi S. To incorporate sequential dynamic features in malware detection engines [C] //Proc of the 2012 European Intelligence and Security Informatics Conf. Piscataway, NJ: IEEE, 2012: 46-52

[33] Ahmed F, Hameed H, Shafiq M Z, et al. Using spatio-temporal information in API calls with machine learning algorithms for malware detection [C] //Proc of the 2nd ACM Workshop on Security and Artificial Intelligence. New York: ACM, 2009: 55-62

[34] Park Y, Reeves D, Mulukutla V, et al. Fast malware classification by automated behavioral graph matching [C] // Proc of the 6th Annual Workshop on Cyber Security and Information Intelligence Research. New York: ACM, 2010: 45:1-45:4

[35] Eskandari M, Hashemi S. A graph mining approach for detecting unknown malwares [J]. Journal of Visual Languages & Computing, 2012, 23(3): 154-162

[36] Moskovitch R, Stopel D, Feher C, et al. Unknown malcode detection via text categorization and the imbalance problem [C] //Proc of the 2008 IEEE Int Conf on Intelligence and Security Informatics. Piscataway, NJ: IEEE, 2008: 156-161

[37] Abou-Assaleh T, Cercone N, Keselj V, et al. N-gram-based detection of new malicious code [C] //Proc of the 28th Annual Int Computer Software and Applications Conf. Piscataway, NJ: IEEE, 2004: 41-42

[38] Moskovitch R, Feher C, Tzachar N, et al. Unknown malcode detection using opcode representation [C] //Proc of the 1st European Conf on Intelligence and Security Informatics. Berlin: Springer, 2008: 204-215

[39] Saxe J, Berlin K. Deep neural network based malware detection using two dimensional binary program features [C] //Proc of the 10th Int Conf on Malicious and Unwanted Software (MALWARE). Piscataway, NJ: IEEE, 2015: 11-20

[40] Mathew J, Kumara M A A. API call based malware detection approach using recurrent neural network—LSTM [C] //Proc of the Int Conf on Intelligent Systems Design and Applications. Berlin: Springer, 2018: 87-99

[41] Chai Yuhan, Qiu Jing, Su Shen, et al. LGMal: A joint framework based on local and global features for malware detection [C] //Proc of the 2020 Int Wireless Communications and Mobile Computing (IWCMC). Piscataway, NJ: IEEE, 2020: 463-468

[42] Oliveira A, Sassi R J. Behavioralmalware detection using deep graph convolutional neural networks [J/OL]. [2020-10-21]. https://www.techrxiv.org/articles/preprint/Behavioral__Malware__Detection__Using__Deep__Graph__Convolutional__Neural__Networks/10043099

[43] Tobiyama S, Yamaguchi Y, Shimada H, et al. Malware detection with deep neural network using process behavior [C] //Proc of the 40th IEEE Annual Computer Software and Applications Conf (COMPSAC). Piscataway, NJ: IEEE, 2016: 577-582

[44] Shukla S, Kolhe G, PD S M, et al. RNN-based classifier to detect stealthy malware using localized features and complex symbolic sequence [C] //Proc of the 18th IEEE Int Conf On Machine Learning And Applications (ICMLA). Piscataway, NJ: IEEE, 2019: 406-409

[45] He Ke, Kim D S. Malware detection with malware images using deep learning techniques [C] //Proc of the 18th IEEE Int Conf On Trust, Security and Privacy in Computing and Communications/13th IEEE Int Conf On Big Data Science and Engineering (TrustCom/BigDataSE). Piscataway, NJ: IEEE, 2019: 95-102

[46] Tahan G, Rokach L, Shahar Y. Mal-id: Automatic malware detection using common segment analysis and meta-features [J]. The Journal of Machine Learning Research, 2012, 13 (1): 949-979

[47] Zhang Jixin, Zhang Kehuan, Qin Zheng, et al. Sensitive system calls based packed malware variants detection using principal component initialized MultiLayers neural networks [J]. Cybersecurity, 2018, 1(1): 1-13

[48] Siddiqui M, Wang M C, Lee J. Detecting trojans using data mining techniques [C] //Proc of the Int Multi Topic Conf. Berlin: Springer, 2008: 400-411

[49] Raff E, Nicholas C. A Survey of machine learning methods and challenges for windows malware classification [J]. arXiv preprint, arXiv:2006.09271, 2020

[50] Lo C T D, Pablo O, Carlos C M. Towards an effective and efficient malware detection system [C] //Proc of the 2016 IEEE Int Conf on Big Data. Piscataway, NJ: IEEE, 2016: 3648-3655

[51] Raman K. Selecting features to classify malware [J/OL]. InfoSec Southwest,[2020-10-21]. https://www.semanticscholar.org/paper/Selecting-Features-to-Classify-Malware-Raman/3100cab4391d933bd7cce4047ce9e67cb1960750

[52] Firdausi I, Erwin A, Nugroho A S. Analysis of machine learning techniques used in behavior-based malware detection [C] //Proc of the 2nd Int Conf on Advances in Computing, Control, and Telecommunication Technologies. Piscataway, NJ: IEEE, 2010: 201-203

[53] Bishop C M. Neural Networks for Pattern Recognition [M]. Oxford: Oxford University Press, 1995

[54] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks [J]. Communications of the ACM, 2017, 60(6): 84-90

[55] Goodfellow I, Bengio Y, Courville A, et al. Deep Learning [M]. Cambridge: MIT Press, 2016

[56] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks [J]. arXiv preprint, arXiv: 1609.02907, 2016

[57] Anderson H S, Roth P. EMBER: An open dataset for training static pe malware machine learning models [J]. arXiv preprint, arXiv:1804.04637, 2018

[58] Ronen R, Radu M, Feuerstein C, et al. Microsoft malware classification challenge [J]. arXiv preprint, arXiv: 1802.10135, 2018

[59] Roberts J M. Virus share[DB/OL].[2020-10-25]. https://virusshare.com/

[60] Georgia I S C T. Open Malware[DB/OL].[2020-10-25]. http://malwarebenchmark.org/

[61] Aghakhani H, Gritti F, Mecca F, et al. When malware is packin'heat; limits of machine learning classifiers based on static analysis features [C] //Proc of the 27th Annual Network and Distributed Systems Security Symp (NDSS 2020). San Diego, CA: ISOC, 2020: 23-26

[62] Zhu Shuofei, Shi Jianjun, Yang Limin, et al. Measuring and modeling the label dynamics of online anti-malware engines [C] // Proc of the 29th USENIX Security Symp. Berkeley, CA: USENIX Association, 2020: 2361-2378

[63] Berlin K, Slater D, Saxe J. Malicious behavior detection using windows audit logs [C] //Proc of the 8th ACM Workshop on Artificial Intelligence and Security. New York: ACM, 2015: 35-44

[64] Íncer Romeo Í, Theodorides M, Afroz S, et al. Adversarially robust malware detection using monotonic classification [C] //Proc of the 4th ACM Int Workshop on Security and Privacy Analytics. New York: ACM, 2018: 54-63

[65] Kolosnjaji B, Zarras A, Webster G, et al. Deep learning for classification of malware system call sequences [C] //Proc of the Australasian Joint Conf on Artificial Intelligence. Berlin: Springer, 2016: 137-149

[66] Song Wei, Li Xuezixiang, Afroz S, et al. Automatic generation of adversarial examples for interpreting malware classifiers [J]. arXiv preprint, arXiv:2003.03100, 2020

[67] Garfinkel T, Adams K, Warfield A, et al. Compatibility is not transparency: VMM detection myths and realities [C] // Proc of the HotOS. Berkeley, CA: USENIX Association, 2007: 7-9

[68] Lindorfer M, Kolbitsch C, Comparetti P M. Detecting environment-sensitive malware [C] //Proc of the 14th Int Workshop on Recent Advances in Intrusion Detection. Berlin: Springer, 2011: 338-357

[69] Raffetseder T, Kruegel C, Kirda E. Detecting system emulators [C/OL] //Proc of the 10th Int Conf on Information Security. Berlin: Springer, 2007 [2020-10-25]. https://link.springer.com/chapter/10.1007/978-3-540-75496-1_1

[70] Peng Fei, Deng Zhui, Zhang Xiangyu, et al. X-force: Force-executing binary programs for security applications [C] // Proc of the 23rd USENIX Security Symp. Berkeley, CA: USENIX Association, 2014: 829-844

[71] Brumley D, Hartwig C, Kang M G, et al. BitScope: Automatically dissecting malicious binaries [R]. Pittsburgh: School of Computer Science, Carnegie Mellon University, 2007

[72] Rossow C, Dietrich C J, Grier C, et al. Prudent practices for designing malware experiments: Status quo and outlook [C] //Proc of the 2012 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2012: 65-79

[73] Egele M, Scholte T, Kirda E, et al. A survey on automated dynamic malware-analysis techniques and tools [J]. ACM Computing Surveys, 2008, 44(2): 1-42

[74] Graziano M, Leita C, Balzarotti D. Towards network containment in malware analysis systems [C] //Proc of the 28th Annual Computer Security Applications Conf. New York: ACM, 2012: 339-348

[75] Kreibich C, Weaver N, Kanich C, et al. Gq: Practical containment for measuring modern malware systems [C] // Proc of the 2011 ACM SIGCOMM Conf on Internet Measurement Conf. New York: ACM, 2011: 397-412

[76] Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks [J]. arXiv preprint, arXiv: 1312.6199, 2013

[77] Kolosnjaji B, Demontis A, Biggio B, et al. Adversarial malware binaries: Evading deep learning for malware detection in executables [C] //Proc of the 26th European Signal Processing Conf (EUSIPCO). Piscataway, NJ: IEEE, 2018: 533-537

[78] Kreuk F, Barak A, Aviv-Reuven S, et al. Adversarial examples on discrete sequences for beating whole-binary malware detection [J]. arXiv preprint, arXiv: 1802.04528, 2018

[79] Demetrio L, Biggio B, Lagorio G, et al. Explaining vulnerabilities of deep learning to adversarial malware binaries [J]. arXiv preprint, arXiv: 1901.03583, 2019

[80] Liu Xinbo, Zhang Jiliang, Lin Yaping, et al. Atmpa: Attacking machine learning-based malware visualization detection methods via adversarial examples [C] //Proc of the 27th IEEE/ACM Int Symp on Quality of Service (IWQoS). Piscataway, NJ: IEEE, 2019: 38: 1-38: 10

[81] Pappas V, Polychronakis M, Keromytis A D. Smashing the gadgets: Hindering return-oriented programming using in-place code randomization [C] //Proc of the 2012 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2012: 601-615

[82] Anderson H S, Kharkar A, Filar B, et al. Learning to evade static PE machine learning malware models via reinforcement learning [J]. arXiv preprint, arXiv: 1801.08917, 2018

[83] Rosenberg I, Shabtai A, Rokach L, et al. Generic black-box end-to-end attack against state of the art API call based malware classifiers [C] //Proc of the 21st Int Symp on Research in Attacks, Intrusions, and Defenses. Berlin: Springer, 2018: 490-510

[84] Hu Weiwei, Tan Ying. Generating adversarial malware examples for black-box attacks based on gan [J]. arXiv preprint, arXiv: 1702.05983, 2017

[85] Rosenberg I, Shabtai A, Elovici Y, et al. Query-efficient black-box attack against sequence-based malware classifiers [J]. arXiv preprint, arXiv: 1804.08778, 2018

[86] Fleshman W, Raff E, Zak R, et al. Static malware detection & subterfuge: Quantifying the robustness of machine learning and current anti-virus [C] //Proc of the 13th Int Conf on Malicious and Unwanted Software (MALWARE). Piscataway, NJ: IEEE, 2018: 18503623: 1-18503623: 10

Wang Jialai, born in 1997. Master candidate in the Institute for Network Sciences and Cyberspace, Tsinghua University. His main research interests include malware analysis, AI security.

汪嘉来, 1997 年生. 硕士研究生. 主要研究方向为恶意软件分析和 AI 安全等.

Zhang Chao, born in 1986. PhD, associate professor in Tsinghua University. His main research interests include vulnerability discovery, program analysis, AI security, malware analysis.

张　超, 1986 年生. 博士, 副教授. 主要研究方向为漏洞挖掘、程序分析、AI 安全和恶意软件分析等.

Qi Xuyan, born in 1984. Master, associate professor. Her main research interests include malware analysis, device vulnerability discovery.

戚旭衍, 1984 年生. 硕士, 副教授. 主要研究方向为恶意软件分析与设备漏洞挖掘.

Rong Yi, born in 2001. Undergraduate student in the School of Software, Tsinghua University. Her main research interest is malware analysis.

荣　易, 2001 年生. 本科生. 主要研究方向为恶意软件分析.