计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2021.20210620
58(10):2140-2162,2021

基于深度学习的软件安全漏洞挖掘

顾绵雪[1,2] 孙鸿宇[2,3] 韩 丹[1,2] 杨 粟[2] 曹婉莹[2] 郭 祯[1] 曹春杰[1] 王文杰[2] 张玉清[1,2,3]
[1](海南大学网络空间安全学院 海口 570228)
[2](国家计算机网络入侵防范中心(中国科学院大学) 北京 101408)
[3](西安电子科技大学网络与信息安全学院 西安 710126)
(gumx@nipc.org.cn)

Software Security Vulnerability Mining Based on Deep Learning

Gu Mianxue[1,2], Sun Hongyu[2,3], Han Dan[1,2], Yang Su[2], Cao Wanying[2], Guo Zhen[1], Cao Chunjie[1], Wang Wenjie[2], and Zhang Yuqing[1,2,3]
[1](*College of Cyberspace Security, Hainan University, Haikou* 570228)
[2](*National Computer Network Intrusion Protection Center (University of Chinese Academy of Sciences), Beijing* 101408)
[3](*College of Cyber Engineering, Xidian University, Xi'an* 710126)

Abstract The increasing complexity of software and the diversified forms of security vulnerabilities have brought severe challenges to the research of software security vulnerabilities. Traditional vulnerability mining methods are inefficient and have problems such as high false positives and high false negatives, which have been unable to meet the increasing demands for software security. At present, a lot of research works have attempted to apply deep learning to the field of vulnerability mining to realize automated and intelligent vulnerability mining. This review conducts an in-depth investigation and analysis of the deep learning methods applied to the field of software security vulnerability mining. First, through collecting and analyzing existing research works of software security vulnerability mining based on deep learning, its general work framework and technical route are summarized. Subsequently, starting from the extraction of deep features, security vulnerability mining works with different code representation forms are classified and discussed. Then, specific areas of deep learning based software security vulnerability mining works are discussed systematically, especially in the field of the Internet of Things and smart contract security. Finally, based on the summary of existing research works, the challenges and opportunities in this filed are discussed, and the future research trends are presented.

Key words deep learning; vulnerability mining; code representation; IoT security; smart contract security

摘 要 软件的高复杂性和安全漏洞的形态多样化给软件安全漏洞研究带来了严峻的挑战.传统的漏洞挖掘方法效率低下且存在高误报和高漏报等问题,已经无法满足日益增长的软件安全性需求.目前,大量的研究工作尝试将深度学习应用于漏洞挖掘领域,以实现自动化和智能化漏洞挖掘.对深度学习应用

收稿日期:2021-06-10;**修回日期**:2021-09-03
基金项目:国家自然科学基金项目(U1836210);海南省重点研发计划项目(ZDYF202012)
This work was supported by the National Natural Science Foundation of China (U1836210) and the Key Research and Development Program of Hainan Province (ZDYF202012).
通信作者:张玉清(zhangyq@nipc.org.cn)

于安全漏洞挖掘领域进行了深入的调研和分析.首先,通过梳理和分析基于深度学习的软件安全漏洞挖掘现有研究工作,概括其一般工作框架和技术方法;其次,以深度特征表示为切入点,分类阐述和归纳不同代码表征形式的安全漏洞挖掘模型;然后,分别探讨基于深度学习的软件安全漏洞挖掘模型在具体领域的应用,并重点关注物联网和智能合约安全漏洞挖掘;最后,依据对现有研究工作的整理和总结,指出该领域面临的不足与挑战,并对未来的研究趋势进行展望.

关键词 深度学习;漏洞挖掘;代码表征;物联网安全;智能合约安全

中图法分类号 TP391

信息技术的高速发展极大地改变了人们的生活方式,便捷的计算机应用程序丰富了人们的生活.近年来,随着计算机软件系统的复杂性增强,潜在的安全漏洞数量呈现递增趋势.美国国家漏洞数据库(National Vulnerability Database,NVD)历年披露的安全漏洞数量[1]如图1所示,从2018年开始,连续3年披露的安全漏洞记录数目均已突破1.5万条大关.

尽管部分披露的软件安全漏洞已经被修复,这并不意味着计算机用户在使用软件系统时所面临的危害有所降低.例如,2014年4月披露的"Shellshock"漏洞①,攻击者利用僵尸网络进行分布式拒绝服务(distributed denial of service, DDoS)攻击,通过搭载基于公共网关接口(common gateway interface, CGI)的Web服务器、OpenSSH服务器或DHCP客户端在受攻击的Bash上执行任意代码,从而在未授权的情况下访问计算机系统.之后,2017年5月出现的"WannaCry"勒索病毒软件②,攻击者利用美国国家安全局(National Security Agency, NSA)在同年3月披露的危险漏洞"永恒之蓝EternalBlue"攻击脆弱的Windows操作系统,入侵用户主机并索要比特币.该勒索病毒波及至少150个国家和地区,给政府、企业和高校等行业造成数以亿计的损失,俨然是一场全球性的互联网灾难.近年来,软件安全漏洞不仅在数量上逐年激增,其形态也表现出复杂性和多样性的特点,给软件系统的正常运行带来了严峻的挑战.

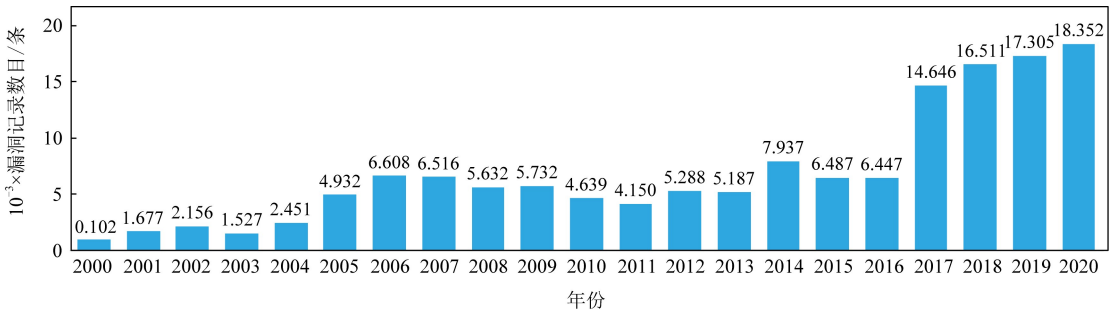

Fig. 1 The number of disclosed vulnerabilities in NVD over the years

图1 美国国家漏洞数据库(NVD)历年披露的漏洞记录数目

目前,学术界和工业界尚未对软件安全漏洞的定义形成统一广泛的共识,本文在总结文献[2-4]关于安全漏洞定义的基础之上,参照文献[5]对软件安全漏洞的定义,即安全漏洞是指在信息产品、信息系统、信息技术在软件生命周期中,软件设计者在需求、设计、编码、配置和运行等阶段有意或者无意产生的软件缺陷,从而使得攻击者在未经授权的情况下访问计算机资源.这些软件缺陷一旦被恶意的攻击者利用,比如权限越级、软件用户隐私数据泄露等,将会导致软件系统之上的正常服务行为偏离,危害信息系统的机密性(confidentiality)、完整性(integrity)和可用性(availability).由于软件安全漏洞隐蔽存在于软件生命周期的各个阶段,如何利用各种技术手段尽早挖掘潜在的安全漏洞,降低对软件系统的危害,是网络空间安全领域的热点研究问题之一.

软件安全漏洞挖掘是安全研究人员检查和分析软件系统中潜在的安全漏洞的主要技术手段.通过利用各种检测工具对软件、源代码以及代码补丁进行

① https://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability

② http://www.cert.org.cn/publish/main/9/2017/20170513170143329476057/20170513170143329476057_.html

审计,或者运行可执行文件对软件的执行过程进行测试,查找其软件缺陷.早期的安全漏洞挖掘技术主要分为静态分析技术、动态分析技术和混合分析技术.静态分析技术是指在不运行程序的情况下,对程序源代码或字节码的语法、语义、控制流和数据流进行分析,从而检测目标程序中可能潜在的安全漏洞.静态分析技术主要包括基于规则的分析技术[6]、二进制对比技术[7]、静态符号执行技术[8]和静态污点分析技术[9]等.静态分析技术不需要运行程序,能够高效快速地完成对大量程序代码的审计,代码覆盖率较高.但随着软件复杂性的增加,依靠人工专家提取漏洞规则常常具有主观性,且构造成本过高,不可避免地导致漏洞误报率和漏报率较高.动态分析技术是指在程序运行情况下,对运行程序的运行状态、执行路径和寄存器状态进行分析,从而发现动态调试器中存在的安全漏洞.动态分析技术主要包括模糊测试[10-11]、动态符号执行技术[12-14]和动态污点分析技术[15]等.动态分析技术一般应用于软件的测试运行阶段,能够从运行程序的状态中追踪程序的执行路径和数据流向,从而提取漏洞特征信息,以提升软件漏洞挖掘的准确率.但动态分析技术程序存在路径覆盖率较低和路径爆炸问题,且需要消耗大量的计算资源[16].混合分析技术是指同时结合静态分析和动态分析技术,以对目标程序进行安全漏洞挖掘.混合分析技术主要依赖于安全研究人员分析程序源代码或字节码的静态特征和运行程序得到的动态特征,或利用动态分析对静态分析的结果进行校验,使得安全漏洞挖掘的准确率提升,从而降低静态分析的高漏报率和增加动态分析的代码覆盖率[17-18].

近年来,随着人工智能(artificial intelligence, AI)技术的兴起,利用 AI 可以自动化地从复杂高维数据中提取数据的有效特征,已经被广泛应用于图像识别[19]、目标检测[20]和自然语言处理[21]等领域.目前,将 AI 应用于安全漏洞挖掘领域主要是利用机器学习(machine learning, ML)、自然语言处理(natural language processing, NLP)和深度学习(deep learning, DL),以实现软件安全漏洞的自动化和智能化研究.

基于机器学习的软件安全漏洞挖掘工作一直受到安全研究人员和软件供应商的关注和重视.在实际研究中,将机器学习技术应用于安全漏洞挖掘领域主要可以包括基于软件代码度量、基于代码属性、基于代码相似性以及基于代码模式的安全漏洞挖掘模型[22-23].

具体而言,软件代码度量是对软件一些特征信息的量化表示,用作对软件质量的度量指标.常用的软件度量指标有开发者活动(developer activities)、复杂度(complexity)、代码变化(code churn)、继承深度(inheritance depth)、耦合度(coupling)和内聚度(cohesion)等.基于软件代码度量的漏洞挖掘模型[24-29]通过选取软件的若干个度量指标量化程序特征来进行表示,在一定程度上能够体现程序的整体属性特征,检测速度较快,但是量化的程序特征信息与漏洞代码本身关联性不强,细粒度不够,只能提供辅助性的漏洞判断,且具有较高的误报率和漏报率.

基于代码属性的漏洞挖掘模型在软件代码度量的基础之上,针对具体的漏洞信息特征,从代码级别挖掘程序代码本身的特征信息.基于代码属性的漏洞挖掘工作[30-37]主要对 Web 端安全漏洞进行研究,应用机器学习算法进行漏洞挖掘,能够检测出 SQL 注入、跨站点脚本攻击(cross-site scripting, XSS)、远程代码执行(remote code execution, RCE)和缓冲区溢出(buffer overflow, CWE-119)等漏洞.基于代码相似性的漏洞挖掘模型在对安全专家手工定义的特征提取之后,使用机器学习等方法计算并比较特征之间的相似度,从而判断是否属于同一类型漏洞.

基于代码相似性的漏洞挖掘模型[38-40]主要能够检测出代码重复利用引起的漏洞问题,在一定程度上能够提升漏洞检测的准确率.但漏洞特征主要依靠安全专家手工定义,且只能发现已知的漏洞信息,应用比较局限.

基于代码模式的漏洞挖掘模型,也可以理解为基于语法语义的漏洞挖掘模型,具体又可分为基于词法分析的漏洞挖掘模型和基于语法分析的漏洞挖掘模型.基于词法分析的漏洞挖掘模型[41-49]主要采用文本挖掘技术对源代码的标识符、函数名和运算符等进行标记,对提取到的有效信息进行抽象表示,接着经过编码模型进行向量化处理之后,得到供机器学习模型训练的特征集合.基于语法分析的漏洞挖掘模型[50-54]通过静态分析技术对程序源代码的数据流和数据依赖进行更深层次的特征表示,主要依据抽象语法树(abstract syntax tree, AST)、数据流图(data flow graph, DFG)、控制流图(control flow graph, CFG)和程序依赖图(program dependency graph, PDG)等语法语义结构提取特征集合.相比于基于软件代码度量的漏洞挖掘模型,基于代码模式

的漏洞挖掘模型在很大程度上考虑了函数组件和函数控制流之间的联系,兼顾了代码的语法语义信息,且能够抽象出代码中更深层次的特征.

然而,基于传统机器学习的软件安全漏洞挖掘模型依赖于安全专家去定义漏洞特征,且只能挖掘已知的漏洞信息,在实际应用环境中无法挖掘未知的漏洞信息,应用范围比较局限.同时,现有的基于机器学习的软件安全漏洞挖掘模型无法指明与漏洞相关的关键语句或特征,使得难以定位安全漏洞存在的精确位置.

随着深度学习技术的快速发展,越来越多的安全研究人员开始将深度学习技术应用于软件安全漏洞挖掘领域.相比于传统的机器学习技术依赖安全专家定义手工特征,深度学习技术通过构建多样性的神经网络对数据进行训练,使得能够更加自动化和智能化地从复杂数据中提取有效特征信息,以提高软件安全漏洞挖掘的准确率,降低漏洞的误报率和漏报率.因此,本文主要侧重于基于深度学习的软件安全漏洞挖掘工作研究,为此广泛收集并调研了自 2013-01—2021-06 期间来自 IEEE Xplore,ACM Digital Library,SpringerLink 和中国知网(CNKI)等国内外数据库以及著名安全会议(IEEE S&P,USENIX Security,CCS,NDSS 等)收录的现有研究工作,如图 2 所示,并总结和归纳基于深度学习的软件安全漏洞挖掘领域目前已有的研究成果,指出该领域的研究趋势.

本文的主要贡献有 4 个方面:

1) 广泛收集并调研了基于深度学习的软件安全漏洞挖掘领域的现有相关文献,总结了基于深度学习的软件安全漏洞挖掘的一般框架和相关技术方法;

2) 以深度特征表示为切入点,分类阐述和分析基于不同代码表征形式的安全漏洞挖掘模型,并分别指出各表征方式中相关方案的优缺点;

3) 从具体的应用场景出发,分别探讨目前深度学习应用于物联网、区块链智能合约以及其他领域漏洞挖掘的研究进展,并系统进行了对比;

4) 分析当前基于深度学习的软件安全漏洞挖掘领域面临的九大挑战和机遇,并对未来的研究趋势进行展望.

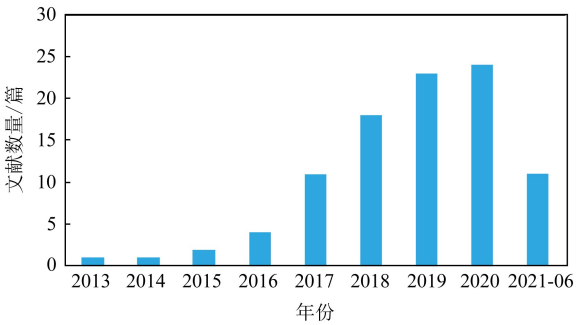

Fig. 2 Literature number of software vulnerability mining based on deep learning

图 2 基于深度学习的软件漏洞挖掘文献数量

1 基于深度学习的软件漏洞挖掘工作框架

通过调研现有的研究工作,我们给出了基于深度学习的软件安全漏洞挖掘模型的一般工作框架,包括数据收集、学习和检测 3 个阶段,如图 3 所示.同时,分析和归纳了现有数码表征和模型学习技术,供读者进一步深入了解基于深度学习的软件安全漏洞挖掘模型的相关技术方法.

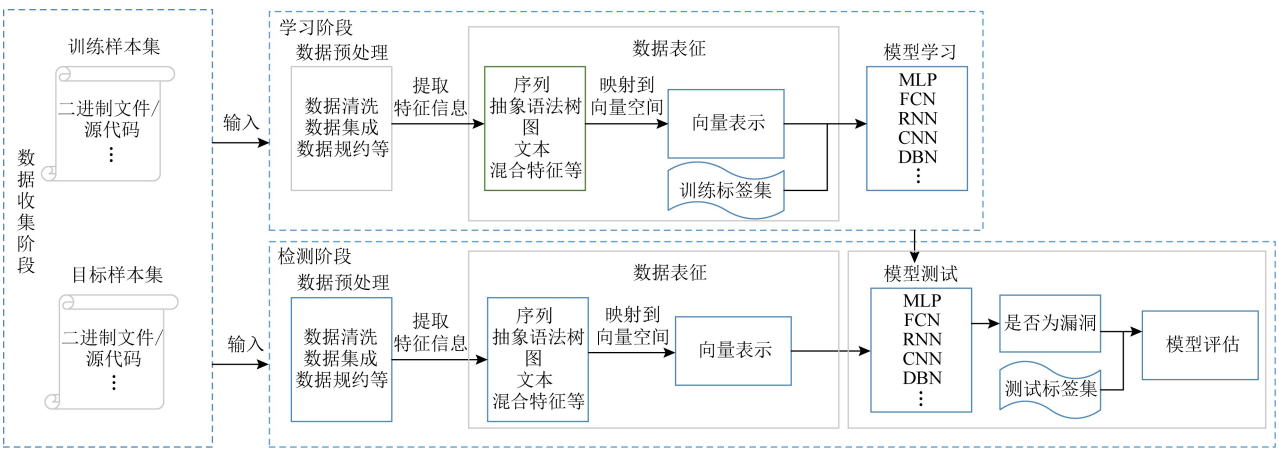

Fig. 3 The framework of software vulnerability mining based on deep learning

图 3 基于深度学习的软件漏洞挖掘工作框架

1.1 数据收集阶段

在数据收集阶段,需要收集大量的漏洞数据供深度学习模型进行训练和学习.通过梳理和分析现有研究工作发现,目前大部分的数据主要来源于NVD,通用漏洞披露数据库(Common Vulnerabilities and Exposures, CVE)、国家信息安全漏洞库(China National Vulnerability Database of Information Security, CNNVD)和 Github 等主流开源网站,且以二进制文件和源代码为主要分析对象.

1.2 学习阶段

学习阶段主要由 3 部分构成,分别是数据预处理、数据表征和模型学习.1)数据预处理阶段.首先要对获取的程序数据集进行预处理,缓解数据重复和数据不平衡问题.一般来说,可以采用数据清洗、数据集成和数据规约等方法对训练数据集进行预处理.2)数据表征阶段.即需要将软件程序数据集解析为合适的表示结构用于模型训练,目前,通常使用序列、抽象语法树、图、文本和混合特征等表征形式抽象出源代码漏洞的特征信息.3)模型学习阶段.由于收集到的软件程序数据集通常是文本表示,并不能直接用于深度神经网络模型进行训练.因此,该部分需要将从数据表征模块抽象出的代码表征映射为向量形式,从而作为训练模型的输入.在多次训练过程中不断调整和优化模型参数,得到一个性能较优的漏洞挖掘模型,并应用于真实数据检测阶段.

1.3 检测阶段

检测阶段中,在获得漏洞挖掘模型之后,可以对目标软件程序进行漏洞预测.检测阶段的流程与学习阶段在数据预处理和数据表征方面类似,对目标程序提取的表征向量化之后,输入到学习阶段得到的漏洞挖掘模型,从而得到预测结果.

1.4 数据表征技术

近年来,大量安全研究人员通过对安全漏洞产生的原理、条件和特征等方面进行深入研究,采用各种数据表征方式和深度学习算法在不同程度上构建了不同的漏洞挖掘模型.由于程序数据包含丰富的特征信息,如何构建合适的数据表征方式最大程度上提取与漏洞相关的特征信息,是一个复杂艰巨的任务.本文通过整理和分析现有研究工作,发现目前代码表征方式主要可以分为 5 类,分别是:基于序列的表征方式、基于抽象语法树的表征方式、基于图的表征方式、基于文本的表征方式和基于混合的表征方式.图 4 给出了基于不同代码表征软件漏洞挖掘文献数量占比情况.

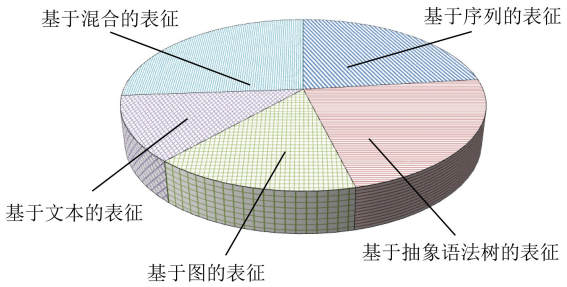

Fig. 4 The percentage of studies based on different code representations for software vulnerability mining

图 4 基于不同代码表征的软件漏洞挖掘研究占比

具体而言,1)基于序列的表征方式对源代码或二进制文件进行词法分析,提取与字符流相关的标识符、函数名和运算符等关键特征信息,同时兼顾执行路径、函数调用序列和语句调用序列等信息.2)基于抽象语法树的表征方式将程序源代码解析为抽象语法树结构并从中提取与树节点相关的语法信息.3)基于图的表征方式则是通过使用图数据结构对源代码的词法和语义属性进行表示,使得能够更加有效地抽象出深层次的代码特征信息.4)基于文本的表征方式则是直接对从程序数据中提取出的特征词进行量化,用于描述和代替程序数据信息.5)基于混合的表征方式通常是融合多种特征表示方式,最大程度上丰富程序数据的特征信息.

如何选择合适的向量编码模型将抽取的特征转换成向量表示形式,在一定程度上将直接影响模型计算的性能.常用的编码模型有 One-hot[55],Word2vec[56]和 Sent2vec[57]等.One-hot[55]编码将文本映射到向量空间,使用 n 维向量对 n 个文本单词进行一一对应编码,但在文本元素过多时,会造成大量的冗余,且无法反映不同元素之间的联系.Word2vec[56]为了克服 One-hot[55]编码的不足,对每个单词分配固定长度的向量,并为语义相似的单词分配距离相近的向量,提升了深度学习模型理解自然语言的能力.Word2vec[56]包含连续词袋模型(continuous bag-of-words, CBOW)和 Skip-Gram 两种模型.其中,CBOW 适合用于小样本数据集,并以周围词作为输入,预测中心词.而 Skip-Gram 适用于数据量较大的情况下,根据中心词,预测其对应的周围词.Sent2vec[57]对 Word2vec[56]中 CBOW 方法进行扩展,以整个句子作为输入,并引入 n-gram,增强语句中单词顺序的嵌入能力.

1.5 模型学习技术

在模型学习阶段,深度学习模型可以实现自动

化提取漏洞特征,且能够获得比“浅层”模型更好的检测性能.本文通过整理和对比现有文献发现,目前应用于漏洞挖掘领域常见的深度学习模型有:多层感知器(multi-layer perception, MLP)、卷积神经网络(convolutional neural network, CNN)、循环神经网络(recurrent neural network, RNN)、长短期记忆网络(long short-term memory network, LSTM)、门控循环单元(gated recurrent unit, GRU)、图神经网络(graph neural network, GNN)、深度置信网络(deep belief network, DBN)以及其他神经网络模型(other deep learning models, Others),如自编码器(auto encoder, AE)、生成对抗网络(generative adversarial network, GAN)等.

其中,MLP 模型在非线性数据上表现较好,但该模型需要大量的训练数据实现拟合,且可解释性不强.CNN 模型可以用来学习结构化的空间数据,但该模型在池化过程中会丢失大量有价值的信息,忽略局部与整体之间的关联性.RNN 模型可以用来处理时序数据,来学习程序数据上下文依赖关系,但在处理序列过长的数据时,容易产生梯度消失问题.LSTM 模型是 RNN 模型的一个变体,在此基础之上添加记忆单元和遗忘门,使得能够捕获序列的长期依赖关系.GRU 模型在将 LSTM 模型的遗忘门、输入门和输出门合并转化为更新门和重置门,以较少的门函数将重要特征进行保存.GNN 模型可以用来学习图中节点、边或子图的低维向量空间表示,以获得深层次的程序数据表示.DBN 模型可以对程序数据在不同概念的粒度上进行抽象,在自动化训练过程中通过调节自身的权重值持久化数据之间的依赖关系,具有更好的性能.AE 是一种无监督学习方法,对高维输入信息进行降维、进行表征学习.GAN 包含一个生成器和一个判别器,分别用于自动学习真实的数据分布和正确判别输入数据是来自真实数据还是生成器.

为了使读者宏观上了解各深度学习算法应用于软件漏洞挖掘的研究情况,本文对该领域现有研究工作进行整理和归纳,占比情况具体如图 5 所示.其中,为了有效量化混合模型采用的深度学习算法,本文对其进行了拆分,分别归纳到相应的模型进行统计.例如,文献[58]分别在 CNN 和 RNN 模型上进行量化统计 1 次.

通过对现有基于深度学习的软件漏洞挖掘文献进行梳理和分析,本文发现大部分工作主要从数据表征方式的改进和学习模型的优化 2 个方面提出新的漏洞挖掘方法,且偏向于数据表征方式的改进,这也是本文侧重于以深度特征表示进行研究综述的依据.

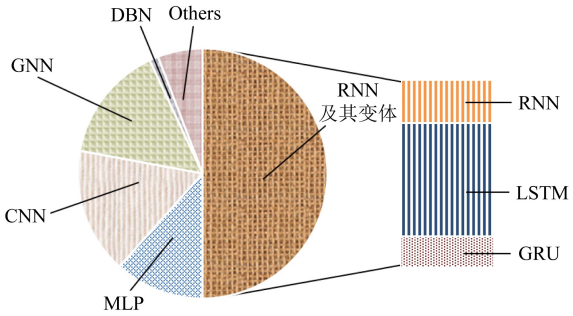

Fig. 5 The percentage of studies which applied different deep learning algorithms for software vulnerability mining

图 5 软件漏洞挖掘模型采用不同深度学习算法研究占比

2 深度特征表示方法

通过整理和分析现有基于深度学习的漏洞挖掘研究工作,目前常见的代码表征方式有序列表征、AST 表征、图表征、文本表征和混合表征.图 6 给出了不同数据表征方式下,现有研究工作文献数量从 2013-01—2021-06 的分布情况.本文对其整理和归纳,以便读者有一个直观的认识.因此,本节将以每种代码表征方式为出发点,分类阐述现有具有代表性的基于深度学习的漏洞挖掘研究工作.同时,在现有研究工作基础之上,本文对每种表征方式的漏洞挖掘模型从不同角度进行讨论和分析,并给出观点,供感兴趣的研究人员对该领域进行进一步研究.

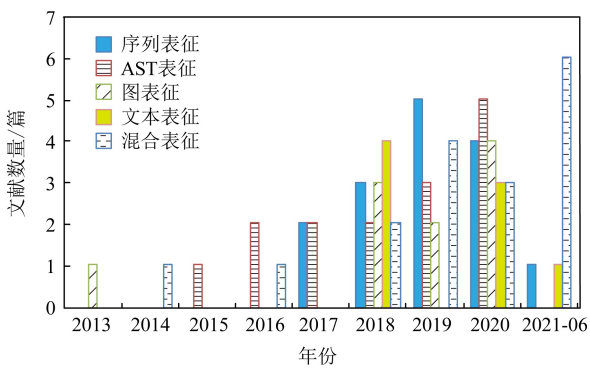

Fig. 6 Literature muber of different code representations for vulnerability mining

图 6 不同代码表征的漏洞挖掘文献数量

2.1 基于序列表征的漏洞挖掘模型

序列表征是指对源代码或二进制文件进行词法分析,提取与字符流相关的标识符、函数名和运算符

等关键特征信息.同时还包含执行路径、函数调用序列和语句调用序列等特征信息.基于序列表征的漏洞挖掘模型[58-72]是通过利用深度神经网络(deep neural network, DNN)自动化提取序列特征信息进行漏洞挖掘.文献[59-60]均从函数调用序列出发,实现漏洞挖掘.文献[59]首次将深度学习应用于序列特征提取,从库/API 函数调用序列出发,采用双向长短期记忆网络(bidirectional long short-term memory network, BLSTM)构建 VulDeePecker 漏洞检测系统.基于启发式方法将程序源代码转化为"code gadget"代码集合,使其产生一组语义联系但不一定连续的多行代码,检测出 4 种未在 NVD 数据库中报告的漏洞信息,具有一定的有效性.然而,VulDeePecker[59]用例漏洞类型较少,误报率较大,且只能给出一段代码中是否包含漏洞信息,无法精确提供与漏洞相关的位置信息.

文献[60]在文献[59]基础上引入控制依赖关系,提出"code attention",以函数调用序列为关键特征信息构建一个多分类神经网络模型 μVulDeePecker,从而辅助系统精确捕捉 40 种漏洞的挖掘工作,具有较高的 F1-指数,并检测出开源软件 Xen 中 2 种未报告的安全漏洞类型.然而,在提取全局特征和局部特征时存在一定的学习偏差,提升了误报率.

文献[59-61]在构建学习模型时,仅使用单深度学习模型进行特征提取.为了对比单深度学习模型和混合深度学习模型的效果,文献[58,62]采用混合深度学习模型进行特征提取.文献[62]分别使用 CNN, LSTM 和混合模型 CNN+LSTM 进行漏洞特征提取,从二进制程序执行过程中收集近万条函数调用序列作为特征用来训练模型.其实验结果发现采用混合神经网络模型在进行特征训练时,往往能够挖掘出更多的特征信息,具有较好的漏洞挖掘效果.

然而,在实际漏洞应用场景中,由于对全局特征和局部特征的学习偏差,准确获取漏洞特征信息并非易事.文献[58]针对文献[63]存在的特征学习偏差问题,采用底层虚拟机中间表示技术(lower level virtual machine intermediate representation, LLVM IR)和混合神经网络模型对源代码关键序列结构信息进行表征,用于自动化漏洞检测.该表征方式能够同时兼顾词法分析并从细粒度上进行漏洞挖掘,能够精确识别出漏洞的具体位置.通过对比不同的单深度学习模型方法,发现基于混合神经网络的漏洞挖掘模型具有较好的性能.

通过调研基于序列表征的漏洞挖掘研究工作发现,文献[58-71]均采用手工标注方式解决样本之间数据不平衡问题,花费了大量的时间成本,且具有较高的误报率.为了解决二进制软件漏洞检测中高误报率和数据不平衡问题,文献[72]结合内核方法和双向循环神经网络(bidirectional recurrent neural network, BRNN)构建深度代价敏感内核机模型(deep cost-sensitive kernel machine, DCKM),用于处理机器指令集序列.该研究将多种数据集进行切分,分别与 6 种开源软件进行对比,其实验结果表明结合深度学习的代价敏感内核机模型能够有效解决样本之间数据不平衡问题,降低误报率.同时,本文发现该工作在多源数据集漏洞收集方面,能够对数据集自动化标注有一定的借鉴意义.

基于序列表征的漏洞挖掘模型利用深度神经网络自动化提取序列特征信息,本文从现有的基于序列表征的漏洞挖掘研究工作中,挑选和总结了 5 项具有代表性的研究工作,具体如表 1 所示.表 1 分别从分析对象、模型构造、检测细粒度、漏洞类型以及性能多个角度进行分析和讨论,并给出了基于序列表征的漏洞挖掘模型领域的一些观点.

Table 1 Comparisons of Some Reviewed Works Which Applied Sequence-based Feature Representation for Vulnerability Mining

表 1 基于序列表征的漏洞挖掘模型部分工作对比

文献	分析对象	模型构造	检测细粒度	漏洞类型	性能	
					准确率/%	F1-指数/%
文献[59]	C/C++源代码	BLSTM	过程内、过程间	CWE-119,CWE-399		95.0
文献[60]	C/C++源代码	BLSTM	过程内、过程间	不限		94.22
文献[62]	二进制代码	CNN,LSTM,CNN+LSTM	程序层次	不限	83.6	83.3
文献[58]	C 源代码	CNN+RNN	函数层次	不限	99.0	98.6
文献[72]	二进制代码	BRNN	程序层次	CWE-119,CWE-399	95.1	90.3

讨论 1. 由表 1 可知,从分析对象而言,对 C/C++源代码以及二进制代码中存在的漏洞挖掘是目前的研究热点.从采用的深度学习模型方面而言,相比于采用单深度学习模型[59-61],利用混合模型[58,62]学习序列漏洞特征信息,往往具有较好的漏洞检测能力,在性能方面可见一斑.从检测细粒度上看,由于没有一个规范统一的漏洞数据集,不同漏洞挖掘工作构建的数据集在一定程度上对检测细粒度产生了不同的影响.从漏洞类型上而言,文献[58,61-62]能够实现多种漏洞类型的挖掘.而针对常见的缓冲区溢出类型(CWE-119)和资源管理溢出类型(CWE-399)漏洞挖掘,文献[59]的挖掘效果较优于文献[72],取得了 4.7%的提升.

观点 1. 基于序列表征的漏洞挖掘模型能够直接对代码进行词法分析,并对字符流相关的关键特征信息、执行路径和调用序列等信息进行统计得到序列表征,映射到向量空间作为神经模型的输入.通过对比和归纳现有研究工作,本文发现:1)利用 DNN 学习得到的序列表征与漏洞特征的关联性较强,具有良好的漏洞检测能力;2)相比于单深度学习模型,混合模型学习到的序列特征信息更加丰富,检测能力也相对较强;3)在漏洞挖掘过程中,提取所需的序列特征需要大量的训练数据,再进行向量化输入到深度学习模型之中,检测速度一般较慢.

2.2 基于抽象语法树表征的漏洞挖掘模型

抽象语法树(abstract syntax tree, AST)是程序编译过程中对源代码抽象语法结构的一种树状表现形式,其中每一个树节点代表实际代码的一种语法结构信息[73].基于抽象语法树特征表示的漏洞挖掘模型[74-87]通过使用 Clang,ANTLR 和 Lex 等开源软件将程序源代码文件生成 AST,接着对 AST 节点进行遍历转化为数据流结构,从而提取层次化的特征信息.

文献[74-76]在不同程度上均实现了项目内漏洞预测(within-project vulnerability prediction, WPVP),并取得了不错的性能.文献[74]结合双向门控循环单元(bidirectional gated recurrent unit, BGRU)提出了一种基于 AST 表征的具有可解释性的细粒度漏洞挖掘模型.该模型能够区分不同行和不同语法元素对于漏洞信息的重要性,对漏洞 AST 中节点关键信息进行标记,使得对漏洞的具体位置进行精确定位,达到细粒度的漏洞挖掘.然而,该工作在生成 AST 的过程中时间漫长且容易出现 AST 语义信息爆炸问题,难以应用于规模较大的软件系统,具有一定的局限性.

文献[75]对生成后的 AST 规模如何缩减这方面进行了深入的研究,提出了一种新型切割 AST 神经网络模型 ASTNN,用来捕捉词法和语义信息.该研究在语句级别上,将一个代码片段得到的较大规模的 AST 分割成多个小语句树,采用 BGRU 对生成的代码片段进行训练.他们的研究成果取得了不错的实验性能.但相比于文献[74],该方案只能用于特定的代码克隆漏洞检测,无法实现多类型漏洞检测以及跨项目漏洞检测任务.

在基于序列表征的漏洞挖掘模型中,利用混合模型学习漏洞表征,常常具有较好的漏洞检测能力,文献[76]针对缓冲区溢出类型和资源管理异常类型漏洞,分别使用 CNN 与 LSTM 提取漏洞的全局和局部特征信息,提出一种结合傅里叶变换的深度卷积 LSTM 神经网络模型用于漏洞检测,并利用注意力机制对关键代码特征进行重要性分析,使得模型具有更加良好的解释性.

通过调研现有基于 AST 表征的漏洞挖掘研究工作,本文发现针对跨项目漏洞预测(cross-project vulnerability prediction, CPVP)研究在文献数量上较少.事实上,跨项目漏洞挖掘需要在一个项目上构造漏洞挖掘模型从而实现另一个项目的漏洞挖掘,在实际开发场景之中,由于 AST 生成规模较大和容易出现语义爆炸等问题,常常导致模型的性能不佳.

在 CPVP 研究中,针对文献[75]易出现的 AST 语义信息爆炸问题,文献[77]发现结合 DBN 对 AST 语法语义信息进行特征降维,能够有效增强 CPVP 的能力.文献[78]首次将迁移学习(transfer learning, TL)应用于跨项目软件漏洞挖掘,证明发现即使在小数量的数据标签项目中,也能取得不错的检测效果.该研究从 6 个开源软件中收集函数层次的数据集,实验结果表明:无论是在 WPVP 或者在 CPVP 中,将 AST 和 TL 应用于漏洞挖掘具有较好的漏洞检测能力.

为了实现 CPVP 中细粒度的漏洞挖掘,文献[79]在文献[75-76]基础上提出了基于注意力机制的双向长短期记忆网络模型(attention-based bidirectional long short-term memory network, ABLSTM),用于漏洞特征提取,取得了较好的漏洞检测效果.然而在提取词法语义的特征模型中没有太大的变化,因此本文推测基于 AST 表征的漏洞挖掘模型在一定程度上虽然较好地保留了代码的语法语义特征,但在进行漏洞特征提取时效果比较有限.

基于抽象语法树的漏洞挖掘模型能够挖掘源代码层次化的特征信息,本文从基于抽象语法树的漏洞挖掘研究工作中挑选和总结了6项具有代表性的研究工作,具体如表2所示.表2分别从分析对象、模型构造、是否跨项目、漏洞类型和性能等多个方面进行对比和分析.

Table 2 Comparisons of Some Reviewed Works Which Applied AST-based Feature Representation for Vulnerability Mining

表2 基于抽象语法树表征的漏洞挖掘模型部分工作对比

文献	分析对象	模型构造	检测细粒度	是否跨项目	漏洞类型	性能	
						精确率/%	F1-指数/%
文献[74]	C/C++源代码	BGRU	函数级别、语句级别	否	不限	93.29	82.76
文献[75]	C/C++源代码	BGRU	语句级别	否	代码克隆漏洞	98.2	95.5
文献[76]	C/C++源代码	CNN+LSTM	函数级别	否	CWE-119,CWE-399	86.4(CWE-119) 95.4(CWE-399)	90.7(CWE-119) 95.7(CWE-399)
文献[77]	Java源代码	DBN	文件级别	是	不限	63.0	64.1
文献[78]	C/C++源代码	BLSTM	文件级别	是	不限	80.0	
文献[79]	源代码	ABLSTM	函数级别	是	不限	92.3	92.5

讨论2. 本文发现基于AST表征方式的漏洞挖掘模型从分析对象而言主要以C/C++和Java源代码为主,针对其他编程语言漏洞挖掘的研究相对较少.在模型构造方面,采用混合神经网络[76]的漏洞挖掘效果优于单深度学习模型[74-75,77-79].从检测细粒度而言,检测粒度越"细",模型的挖掘性能越高.也就是说,从语句级别[75]进行漏洞特征的提取,会获得比函数级别[76,79]和文件级别[77-78]更好的挖掘效果.从是否能够实现跨项目漏洞挖掘而言,能够发现基于AST表征方式的漏洞挖掘模型在挖掘能力上明显优于基于序列表征方式的漏洞挖掘模型.从漏洞类型上看,基于AST表征方式的漏洞挖掘不仅能够实现某种特定类型漏洞类型[75-76]的挖掘,也能实现多种漏洞类型[74,77-79]的挖掘.通过分析发现,降维技术和迁移学习能够在CPVP中取得不错的效果.

观点2. 基于AST表征的漏洞挖掘模型能够实现对程序源代码的抽象表示,完整地保留程序的语法语义信息,删除了一些与实际语法结构不相关的细节,如程序的注释和分界符号等,适合对程序进行分析.因此,本文发现:1)相比于基于序列表征的漏洞挖掘模型,基于AST表征的漏洞挖掘模型能够完整保留源代码的词法和语义语法信息,检测能力相对较优;2)由于AST规模较大,在生成和提取函数节点时会花费较长的时间成本,检测速度也相对较慢.

2.3 基于图表征的漏洞挖掘模型

基于图特征表示的漏洞挖掘模型[30,88-96]是通过使用图数据结构对源代码的词法和语义属性进行表示,使得能够更加有效地抽象出深层次的代码特征信息.目前常用的图结构有:DFG、CFG、PDG、数据依赖图(data dependency graph, DDG)和代码属性图(code property graph, CPG)[51]等.相比于AST对源代码进行直接表示,DFG是一种结构化系统分析方法,以图形方式表示源代码在系统内部的逻辑流向.CFG则用来描述代码语句的执行顺序,以及程序运行过程中遍历到的所有执行路径.PDG对源代码进行标记的有向多重图,能够反映程序的控制依赖和数据依赖关系.DDG是描述数据之间的相互制约关系,主要分为函数依赖和连接依赖关系.CPG是将程序的CFG和DDG等信息进行结合,从而更好地表征程序的结构信息.

基于图表征的漏洞挖掘模型主要从不同程序源代码或二进制文件中进行安全漏洞挖掘.文献[30,88-89]从语句层次出发进行漏洞挖掘,均取得了不错的性能.文献[30]采用静态代码属性从CFG和DDG提取源代码中与漏洞特征相关的信息,实现Web应用中SQLI和XSS漏洞挖掘.该研究基于图表征对比了不同的机器学习模型和MLP的性能,发现采用MLP的实验结果要比同一数据集上采用不同机器学习模型的实验结果效果好得多.但图的生成过程引入了不必要的重复节点信息,降低了模型的有效性.

文献[88]在数据预处理阶段将数据依赖和控制依赖关系,通过严格的去重步骤,移除了重复编译的特征向量并生成系统依赖图(system dependency graph, SDG),输入到CNN中进行学习得到图表征.SDG由去重后的最小中间代码表示生成而得,发现具有中间表示学习阶段的方法有更好的性能.

然而,文献[30,88]实现了单个函数中语句级别的漏洞检测,无法进行多个函数比对.为了同时实现多个函数比对和在语句级别上的漏洞检测,文献[89]将深度学习与程序切片技术相结合,提出了一种面向二进制代码漏洞检测的深度学习系统 BVDetector.首先对二进制程序的数据流和控制流分析,基于 CFG 提取库/API 函数调用,并基于 PDG 对多个库/API 函数调用生成各自对应的程序切片,采用 BGRU 实现语句级别的漏洞挖掘,提升了漏洞挖掘的效果.文献[30,88-89]都是从语句级别出发构建了不同的深度学习模型用于漏洞挖掘,但表现出来的性能有所不佳.这就表明图表征方式的优势没有完全挖掘出来,应该构建适合挖掘图语法语义信息的深度学习模型,实现挖掘性能的提升.

不少研究工作采用图神经网络(graph neural network, GNN)从代码块级别[90]和函数级别[91]实现漏洞挖掘,取得了不错的效果.文献[90]以 C♯编程语言为分析对象,从代码块级别出发,预测每个代码块中含有的变量名(VARNAMING)和判断变量是否被正确使用(VARMISUSE).他们利用 PDG 边之间的语法和语义信息,采用 GNN 构建漏洞挖掘模型,取得了较好的检测能力.然而文献[30,88-90]仅能挖掘现有已知的漏洞类型,无法检测其他未知类型的漏洞,具有一定的局限性.

文献[91]首次将深度学习技术与漏洞外推(vulnerability extrapolation)概念相结合,研究利用 PDG 提取已知漏洞函数级别的控制依赖和数据依赖关系,采用 GNN 模型进行漏洞模式外推,发现了一些未曾公布的漏洞信息,这就表明将深度学习技术与漏洞外推结合的漏洞挖掘模型具有一定的有效性.然而,由于需要对已知漏洞的特征信息进行全面的深入分析,这就要求安全研究人员构建合适的模型用于学习代码表征,同时漏洞外推只能针对某一种特定漏洞进行挖掘,无法检测其他类型的漏洞,可扩展性不强.

基于图表征的漏洞挖掘模型通过使用图数据结构对源代码特征进行表示,使得能够抽象出深层次的代码特征信息.本文从现有的基于图表征的漏洞挖掘研究工作中,选择其中 5 项具有代表性的研究工作进行了总结和对比,具体情况如表 3 所示.表 3 分别从分析对象、模型构造、检测细粒度、漏洞类型以及性能多个角度进行分析和讨论.

Table 3 Comparisons of Some Reviewed Works Which Applied Graph-based Feature Representation for Vulnerability Mining

表 3 基于图表征的漏洞挖掘模型部分工作对比

文献	分析对象	模型构造	检测细粒度	漏洞类型	性能	
					准确率/%	F1 指数/%
文献[30]	PHP 源代码	MLP	语句级别	SQLI,XSS	92(SQLI) 82(XSS)	
文献[88]	C 源代码	CNN	语句级别	不限		93.0
文献[89]	二进制代码	BGRU	语句级别	内存损坏、数值处理	96.7	89.9
文献[90]	C♯源代码	GNN	代码块级别	VARNAMING, VARMISUSE	53.6(VARNAMING) 85.5(VARMISUSE)	65.8
文献[91]	C/C++源代码	GNN	函数级别	不限	99.2(Training score) 85.2(Test score)	

讨论 3. 文献[30,88-91]在不同程度上构建了基于图表征的漏洞挖掘模型,从分析对象而言,基于图表征方式的漏洞挖掘模型能够实现多种编程语言[30,88,90-91]以及二进制文件[89]的漏洞挖掘.从检测细粒度而言,以基本块属性为检测细粒度的图表征方式[90],检测效果相比于函数级别和语句级别的图表征方式[30,88-89,91]欠佳.从挖掘的漏洞类型来看,基于图表征方式的漏洞挖掘模型在特定的漏洞类型挖掘[30,,89-90]以及多种类型漏洞挖掘[88,91]方面,均取得了不错的检测效果.

观点 3. 基于图的表征方式能够抽象出源代码中词法和语义更深层次的特征信息.但由于生成图表征这一过程复杂性较高,以及构建深度学习模型算法存在时间复杂度和空间复杂度较高的问题.因此,本文发现:1)选择图表征能在一定程度上保留源代码完整的语法和语义信息,能够帮助提升漏洞挖掘的效果,但检测速度较慢,很难应用于大规模的软件系统;2)利用图嵌入技术和图神经网络模型学习图表征,可能会带来更好的漏洞挖掘效果;3)将深度学习与漏洞外推相结合,能够有效挖掘一种特定类型的漏洞,在未来的研究中如何增强现有漏洞外推工作的能力,是一个值得探索的研究课题.

2.4 基于文本表征的漏洞挖掘模型

代码文本是指源代码的表面文本、汇编指令和代码 lexer 处理的源代码.文本特征表示是指对从文本中提取出的特征词进行量化,用于描述和代替文本信息.目前,基于文本特征表示的漏洞挖掘模型[97-107]常使用分词和词频统计等方法对程序源代码进行表征,以提取有效的源代码特征信息.

文献[97-102]采用文本挖掘与深度学习技术结合的方式实现漏洞挖掘,将深度学习应用于程序分析,均取得了不错的检测性能.文献[97]提出了构建程序向量表示的编码标准,利用词频统计方法对 Java 源文件的漏洞模式进行表征,采用单深层全卷积神经网络(fully convolutional networks, FCN)对特征向量进行学习和训练,其实验结果表明深度学习模型能够获得比"浅层"学习模型更好的挖掘性能.事实上,仅依靠词法分析没有考虑到语义的上下文关系,只对程序源代码信息进行了粗糙的语法语义信息提取,限制了漏洞挖掘模型的性能.

文献[99-101]以 C/C++为分析对象,在不同程度上构建了深度学习模型用于漏洞挖掘.文献[99]发现仅使用词法分析得到的漏洞特征[97]具有较低的性能,将 CNN 与 NLP 相结合抽象出候选集样本中的特征,提出了一个系统化的特征提取框架 PreNNsem,用于漏洞挖掘.文献[100]将迁移学习应用于漏洞挖掘,将基于转换器的双向编码表征(bidirectional encoder representation from transform, BERT)方法与 BLSTM 结合,从文件级别出发,实现了从英语文本信息到计算机文件特征提取的过渡,取得了不错的检测性能.然而,由于检测的细粒度只能从文件级别出发,缺少对程序代码语义的信息提取,降低了模型的挖掘性能.

除利用 FCN[97],CNN[99]和 BLSTM[100]从语料库中提取漏洞上下文模式和结构信息之外,文献[101]从语句级别提取语法语义信息,采用神经记忆网络(neural memory network, NMN)将源代码的行编码为特征向量,并存储到内部记忆块中,有效提升了缓冲区溢出漏洞类型挖掘的性能.文献[102]以 PHP 源代码为分析对象,结合深度学习技术和 NLP 解决 PHP Web 应用程序中 SQL 注入漏洞检测问题,丰富了漏洞挖掘的能力.通过分析基于文本表征的漏洞挖掘研究工作发现,大部分研究工作[97-102]仅使用各自构建的数据集进行漏洞挖掘,无法进行基准参考.文献[103]为了对比不同深度学习模型在同一数据集上的表现效果,分别采用 CNN 和 RNN 进行漏洞特征提取,从函数级数据出发,设计了 C/C++词法分析器归一化标识符信息.其实验结果发现在同一数据集上 CNN 表现出比 RNN 更好的性能.

基于文本表征的漏洞挖掘模型直接将源代码作为文本处理,能够有效地提取源代码特征信息.本文从现有的基于文本表征的漏洞挖掘研究工作中,选择其中 6 项具有代表性的研究工作进行了总结和对比,具体情况如表 4 所示.表 4 分别从分析对象、模型构造、检测细粒度、漏洞类型以及性能多个角度进行分析和讨论.

Table 4 Comparisons of Some Reviewed Works Which Applied Text-based Feature Representation for Vulnerability Mining

表 4 基于文本表征的漏洞挖掘模型部分工作对比

文献	分析对象	模型构造	检测细粒度	漏洞类型	性能	
					准确率/%	F1 指数/%
文献[97]	Java 源代码	FCN	文件级别	不限	93.0	
文献[99]	C/C++源代码	CNN	语句级别、单词级别	不限		93.6
文献[100]	C/C++源代码	BERT+BLSTM	文件级别	不限	93.49	
文献[101]	C 源代码	NMN	语句级别	CWE-119	86.0	92.0
文献[102]	PHP 源代码	LSTM	函数级别	SQLI	95.35	
文献[103]	C/C++源代码	CNN,RNN	函数级别	不限	69.8	84.0

讨论 4. 相比于基于序列、基于 AST 以及基于图 3 种表征方式的漏洞挖掘模型,基于文本表征的漏洞挖掘模型采用 DNN 直接将源代码作为文本进行输入,能抽象出隐藏在代码中的语法语义信息.从分析对象而言,基于文本表征的漏洞挖掘模型能够采用不同类型的网络结构从各种输入数据中提取抽象特征,从而挖掘易受攻击的代码片段的语义特征.在模型方面,FCN 可以拟合高度非线性和抽象的模式,比传统机器学习算法更有发展潜力[97].CNN 模型可以用来学习结构化的空间数据,能够辅助安全

研究人员进行漏洞挖掘的工作[99,103].RNN 和它的变形模型能够捕获顺序数据的长期依赖关系,对于理解多种类型漏洞的语义至关重要[100,102-103].最新的研究工作使用 NMN 用来存储具有长期依赖关系的代码文本,取得了不错的效果[101].

观点 4. 从文献数量上来看,将文本挖掘与深度学习技术结合的方法相对较少,针对不同的分析对象、如何构建合适的深度网络模型提取代码的上下文依赖关系和设计统一公开的数据集等方面,基于文本特征表示的漏洞挖掘模型依然是一个值得研究的方向.

2.5 基于混合表征的漏洞挖掘模型

基于混合特征表示的漏洞挖掘模型[108-123]是指结合序列特征表示、抽象语法树特征表示、图特征表示和文本特征表示中至少 2 种特征表示方法,用于源代码词法和语义信息的有效提取.相比于单个特征表示的漏洞挖掘模型,基于混合特征表示的漏洞挖掘模型往往具有较高的性能.

使用单表征方式在一定程度上能够抽象出代码的语法和语义信息,但由于易受攻击的代码模式多种多样,且代码片段之间的上下文依赖关系错综复杂,定义通用描述所有类型的特征集是几乎不可能的一项工作.文献[108-112]采取多个表征形式尽可能完整地提取代码片段特征信息,来进一步弥补语义之间的差距.

早期的混合特征提取方法[50]需要安全研究人员手工进行,无法自动化地进行特征提取.文献[108]面向 Android 二进制文件提取符号特征序列信息和 AST 语义特征共同构建缺陷特征,采用 DNN 来进行缺陷预测,其受试者工作特征曲线(area under curve, AUC)在 WPVP 和 CPVP 中均取得了不错的实验性能.

文献[109-112]使用 AST 和图表示方法共同提取漏洞特征,提高了模型对程序代码的表示能力.文献[109]从函数级数据出发,采用一种复合流动挖掘模型,以 AST 为基础骨架,增加 CFG 和 DFG 用于追踪控制流和数据流的依赖关系.采用 GNN 对图表征进行建模,提升模型对程序代码的理解能力.文献[110]提出了基于语法、基于语义和向量表示的漏洞挖掘框架 SySeVR,侧重于抽象出与过程间与过程内漏洞相关的语法和语义信息的程序表示,应用于多种开源软件产品,检测到了 15 个在 NVD 中未报告的漏洞,进一步证明了该工作的有效性.文献[111]在文献[110]基础之上,引入 LLVM 定位漏洞具体的位置,对比了不同的深度学习模型,表明 BGRU 取得了最佳的检测效果.

一般从直观上认为,提取的特征越复杂,所得到的语法和语义信息越丰富,检测效果会较优.但在文献[112]中提出了一种基于简化代码属性图表征的漏洞挖掘模型,从函数级数据出发,以较少的表征信息最大程度上保留了漏洞的特征信息,选取 GNN 和 MLP 自动学习表征,其实验结果发现该方法用于漏洞挖掘时,具有良好的性能.

基于混合表征的漏洞挖掘模型采取多种特征表示方法,用于提取源代码词法和语义信息.本文从现有的基于混合表征的漏洞挖掘研究工作中,选择其中 5 项具有代表性的研究工作进行了总结和对比,具体情况如表 5 所示.表 5 分别从分析对象、模型构造、检测细粒度、是否跨项目、漏洞类型以及性能多个角度进行分析和讨论.

Table 5 Comparisons of Some Reviewed Work Which Applied Mixed-based Feature Representation for Vulnerability Mining

表 5 基于混合表征的漏洞挖掘模型部分工作对比

文献	分析对象	模型构造	检测细粒度	是否跨项目	漏洞类型	性能		
						曲线下面积/%	准确率/%	F1 指数/%
文献[108]	二进制文件	DNN	语句级别	是	不限	83.08(WPVP) 66.36(CPVP)		
文献[109]	C 源代码	GGNN	函数级别	否	不限		72.26	73.26
文献[110]	C/C++源代码	BGRU	过程内、过程间	否	不限		98.0	92.6
文献[111]	C 源代码	BRNN	过程内、过程间	否	不限		98.8	98.2
文献[112]	C/C++源代码	GNN,MLP	函数级别	否	不限		89.2	83.6

讨论 5. 本文从不同角度分析和归纳了基于混合表征方式的漏洞挖掘模型,从分析对象而言,目前大部分文献主要集中于源代码漏洞检测[109-112],二进制文件漏洞检测[108]文献相对较少.从模型构造而言,BRNN[111]和 BGRU[110]能够取得比其他深度学习模型较优的检测性能.从检测细粒度而言,文献

[110-111]面向过程内和过程间进行漏洞特征提取，能够实现比函数级别[109,112]和语句级别[108]更“细”的语法语义信息提取.文献[108]虽能够实现跨项目漏洞挖掘，但仅限于二进制文件.由于不同编程语言和应用环境的差异性，基于混合特征表示的漏洞挖掘模型在跨项目漏洞挖掘这方面的研究仍然需要安全人员投入大量的时间进行探索.

观点 5. 相比于前 4 种单特征表示方法，基于混合表征的漏洞挖掘模型能够综合考虑程序源代码的词法、语法和结构、语义信息，同时兼顾函数组件与函数控制流之间的依赖关系，与漏洞的特征具有较强的关联性，检测能力也更强.因此，如何融合多种特征实现自动化和细粒度漏洞挖掘，是一个值得探索的研究课题.

2.6 小 结

最大程度上完整保留程序数据的语法语义信息，抽象出代码蕴含的更深层信息，构建合适的数据表征方式，将有助于深度神经网络在进行漏洞挖掘时拥有更好的检测性能.

本节分别从序列表征、AST 表征、图表征、文本表征和混合表征等方面分析了现有一系列基于深度学习的漏洞挖掘模型研究成果，同时指出了现有代表性研究工作的优缺点，并给出了一些研究思路.其工作发展历程如图 7 所示：

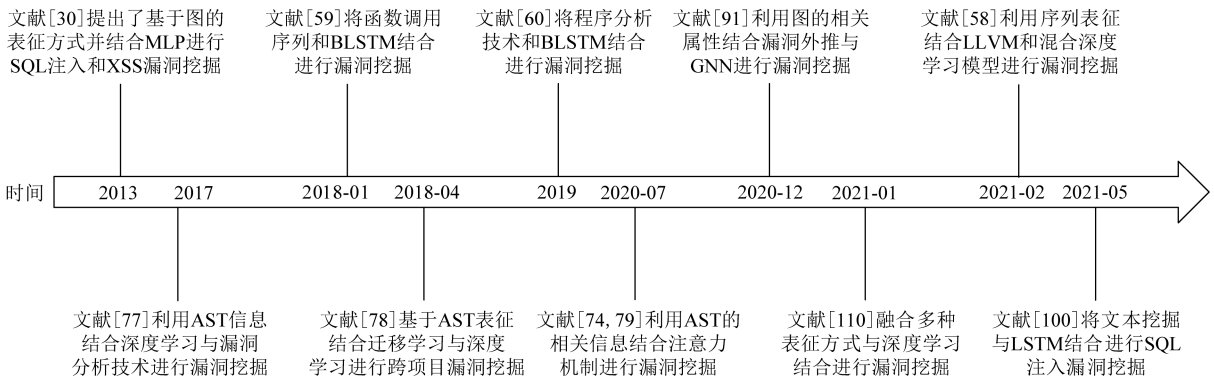

Fig. 7 The development of software vulnerability mining based on deep learning

图 7 基于深度学习的软件漏洞挖掘发展历程

3 具体应用场景的漏洞挖掘模型

通过调研现有研究工作，本文发现在实际应用场景中，基于深度学习的漏洞挖掘模型表现出来的差异有所不同.因此，本节从不同的的应用场景出发，给出基于深度学习的漏洞挖掘模型在物联网、区块链智能合约和其他领域(浏览器漏洞、文档类型漏洞和操作系统安全漏洞)漏洞挖掘的研究进展.

本文选择物联网、区块链智能合约这 2 个应用领域进行详细分析和讨论，有 3 个原因.1)通过梳理现有文献发现，现有结合深度学习进行漏洞挖掘的研究成果数量主要集中于这 2 个领域，有必要分别进行单独调研和分析.2)物联网设备底层的第三方库/API 源代码往往存在大量的安全缺陷，一旦其漏洞被恶意的攻击者进行利用，会造成整个网络空间安全的稳定性.同时，区块链应用场景的多样化加剧了智能合约的复杂性，智能合约作为区块链的核心部分，往往都是以公开透明的方式存在于区块链中，一旦出现安全漏洞，将会带来不可估量的损失，危害系统的安全.3)将深度学习应用于其他领域的研究成果在数量上相对较少，因此本文将其归纳到其他领域中进行探讨，以供读者全方位了解深度学习在具体应用场景的研究进展.

3.1 物联网中基于深度学习的漏洞挖掘模型

物联网是近几年互联网时代研究的热潮之一，大量智能产品给人们带来便捷的同时，其安全事件也不断攀升.因此，针对物联网中智能产品中的安全漏洞问题，如何利用深度学习技术实现智能化和自动化漏洞挖掘迫在眉睫.

需要注意的是，本文主要侧重于物联网中基于第三方库/API 中存在的安全漏洞，对采用相似度检测方法或者二进制关联算法实现跨平台二进制代码安全漏洞检测[124-129]进行对比和分析.

文献[124-126]以基本块为检测细粒度，对二进制代码构建了不同的深度学习模型，用于漏洞检测.文献[124]发现采用图表征提取漏洞特征时，采用图嵌入神经网络(graph embedding neural network,

GENN)增强了漏洞挖掘的效果.实验结果表明:该方法能够应用于不同的应用场景,但依赖于手工提取漏洞特征,检测速度相对较慢且准确率较低.文献[124]是利用 GENN 对二进制代码相似性检测的一次次有效尝试.文献[125-126]将 CFG 和 DFG 相结合,形成标记语义流图(labeled semantic flow graph, LSFG),并且采用深度学习算法实现跨平台二进制相似性漏洞搜索工具.通过与现有研究工作[124]比较,体现出较好的函数语义和搜索精度.然而文献[124-126]仅仅粗浅以代码块为检测细粒度,并没有考虑函数之间的依赖关系,降低了模型的检测性能.

文献[127]从函数级别出发,兼顾函数之间的依赖关系,测试多种无监督学习计算属性控制流图,自动化提取跨平台特征,能够提升检测的 AUC 性能.文献[128]认为 Gemini[124]手工提取和压缩特征的过程会损失语义信息,发现在不同平台编译出的二进制代码的控制流图节点顺序非常相似,采用 BERT 和 CNN 提取语义信息,取得了较好的效果.然而由于在检测细粒度上仍以基本块和函数级别为主,不同文献所构建的深度学习模型的检测性能无法进一步改善性能.为了更加细粒度的挖掘语法语义信息,文献[129]分别从函数过程内和过程间进行特征提取,提出了新的跨平台二进制代码相似性检测方案 αDiff,采用 CNN 实现跨平台的漏洞挖掘,取得了不错的实验结果.

本文从现有的针对物联网安全漏洞挖掘问题的研究工作中,选择其中 6 项具有代表性的研究工作进行了总结和对比,具体情况如表 6 所示:

Table 6 Comparisons of Some Reviewed Works Based on Deep Learning for Vulnerability Mining in IoT

表 6 物联网中基于深度学习的漏洞挖掘部分工作对比

文献	表征方式	模型构造	检测细粒度	是否跨平台	性能			
					召回率/%	准确率/%	AUC/%	效率
文献[124]	图	GENN	基本块	是		80		较高
文献[125]	图	GENN	基本块	是		81.3		较高
文献[126]	图	GENN	基本块	是		88.89		较高
文献[127]	图	GENN	函数级别	是			95.6	高
文献[128]	图	BERT+CNN	函数级别	是		88.41		高
文献[129]	图	CNN	过程内、过程外	是	99.6			高

表 6 分别从表征方式、模型构造、检测细粒度、是否跨项目以及性能多个角度进行分析和讨论,并且从召回率、准确率、AUC 和效率等方面进行了简单的评估.

讨论 6. 本文发现物联网中基于第三方库/API 中存在的安全漏洞检测主要采用图表征方式提取特征,采用 GENN[124-127]和 CNN[128-129]等主流深度神经网络模型进行训练,文献[129]从函数过程内和过程间出发,实现比函数级别[127-128]和基本块属性[124-126]更丰富的漏洞特征信息提取,取得了更好的检测效果.

观点 6. 目前,针对物联网中第三方库/API 中存在的安全漏洞检测,主要采用图表征方式进行漏洞特征表示,虽然能取得一定的检测效果,但检测效果提升不是很明显.由此看来,改善现有图表征方式,进一步丰富漏洞特征信息或者构建其他新的代码表征方式,挖掘蕴含在代码中更深层面的信息,在未来的漏洞挖掘研究中值得探索.总体来说,利用深度学习进行物联网第三方库/API 中的安全漏洞检测研究目前还在起步阶段,是一个值得安全研究人员探讨和实践的方向.

3.2 智能合约中基于深度学习的漏洞挖掘模型

智能合约是区块链上可执行合约条款的计算机交易协议,区块链通过智能合约向链上用户提供复杂多样的业务功能,但智能合约的复杂性会随着应用场景的多样化不断增加,势必会造成大量安全漏洞存在,给整个计算机系统带来巨大的威胁.

智能合约安全漏洞检测面临一系列的安全挑战.一方面,由于区块链中许多项目大都会公开智能合约代码,这就使得在提升用户对部署合约信任度的同时也降低了黑客攻击的成本.另一方面,区块链技术起步相对较晚,发展时间短,在其开发过程中自身存在严重的缺陷,严重阻碍了区块链的技术进步发展.因此,针对智能合约漏洞安全检测问题,需要安全研究人员采用相应的安全技术充分分析潜在的安全威胁,尽可能规避漏洞.

为了解决以上问题,目前不少安全研究人员尝试将深度学习引入到智能合约漏洞检测领域[130-135].文献[130-133]均从字节码层面出发,采用不同的深度学习算法进行漏洞挖掘.文献[130]利用深度学习

辅助智能合约漏洞检测,采用 LSTM 在字节码层面分析智能合约的威胁,取得了不错的效果.文献[131]同样从字节码出发,将智能合约的字节码转化为 RGB 颜色,再转化为图像输入到 CNN 自动化提取丰富的特征信息,从而克服安全专家手动定义规则的主观性.该方法将图像相似性识别领域的相关方法应用于智能合约安全漏洞检测,给读者提供了新的思考方向,具有一定的启示作用.

文献[132]采用 DNN 对易受攻击的以太坊虚拟机字节码进行分析,提出了静态分析工具 Eth2Vec 用于提取集成代码和 AST 特征信息,取得了不错的实验结果.然而,该研究只针对部分字节码进行分析,模型的可扩展性不强.考虑到文献[132]在挖掘模型在可扩展性上的不足,文献[133]首先将迁移学习应用于智能合约安全漏洞检测,提出基于深度神经网络的以太坊智能合约安全漏洞检测框架 ESCORT,利用文本表征字节码信息,采用 RNN 实现自动化特征提取,取得了不错的检测效果.

除此之外,文献[134-135]从函数层次出发,利用 GNN 构建漏洞挖掘模型,取得了不错的检测性能.文献[134]构造了智能合约函数的词法和语义结构图,利用图表征方式进行漏洞检测,取得了不错的实验性能.文献[135]将深度学习与模糊测试技术结合,从而生成智能合约中更好的测试用例和交易调用序列.使用符号执行引擎产生覆盖率较高的调用序列,并采用深度神经网络对序列特征进行学习得到训练模型,取得了较高的代码覆盖率.

本文从现有针对智能合约安全漏洞挖掘问题的研究工作中,选择了其中 6 项具有代表性的研究工作进行总结和对比,并给出了智能合约安全漏洞挖掘领域的一些观点.表 7 分别从表征方式、模型构造、检测细粒度、是否跨项目以及性能多个角度进行讨论.

Table 7 Comparisons of Some Reviewed Works Based on Deep Learning for Vulnerability Mining in Smart Contract

表 7 智能合约中基于深度学习的漏洞挖掘部分工作对比

文献	表征方式	模型构造	检测细粒度	是否跨项目	性能			
					覆盖率/%	准确率/%	精确率/%	F1 指数/%
文献[130]	序列表征	LSTM	字节码级别	否		99.57		86.04
文献[131]	图像	CNN	字节码级别	否		97.39		
文献[132]	混合特征	DNN	字节码级别	否			95.6	82.0
文献[133]	文本表征	RNN	字节码级别	是			98.0	95
文献[134]	图	GNN	函数级别	否		84.48		79.19
文献[135]	序列表征	GNN	函数级别	否	92			

讨论 7. 从表征方式而言,将序列表征[130,135]、图表征[134]、文本表征[133]或混合表征[132]等与深度学习技术结合,均取得了不错的检测性能.从检测细粒度而言,文献[130-133]从字节码级数据出发,能够实现比从函数级数据[134-235]出发更深层次的语法语义信息提取,取得了相对较优的性能.从是否能够实现跨项目漏洞挖掘而言,文献[133]将迁移学习应用于智能合约跨项目安全漏洞检测,取得了不错的检测性能.

观点 7. 将深度学习应用于智能合约安全漏洞挖掘还处于研究初期,仍然还有很长的路要走.通过对现有研究工作的对比和分析,本文认为:1)相比于单表征方式的漏洞挖掘模型,基于混合表征的漏洞挖掘模型应用于智能合约安全漏洞挖掘,能够进一步丰富漏洞的特征信息,提升漏洞挖掘的能力,但这方面的研究从文献数量上来看相对较少,未来这方面的工作依然值得探索;2)在跨项目智能合约安全漏洞检测中,迁移学习的能力未有明确的上限,需要进一步进行考量.

3.3 其他领域

将深度学习应用于除了 3.1~3.2 节物联网以及智能合约安全漏洞挖掘领域之外,在浏览器安全漏洞、文档类型漏洞和操作系统安全漏洞挖掘等方面也存在着相关的研究.本节进行集中介绍,以便读者了解在具体应用场景中,应用深度学习进行漏洞挖掘的现有研究工作.

为了缓解操作系统中 x86 程序出现的内存崩溃漏洞,文献[136]提出了一个轻量级的深度学习检测系统 VDiscover,具有良好的性能.文献[137]提出了基于深度学习的通用模糊测试框架 SmartSeed,用于生成有价值的文件作为模糊工具的测试用例,在检测过程中发现了 16 个新的安全漏洞,表明该方法能够有效提升模糊测试触发漏洞的能力.此外,还有针对文档类型漏洞的智能化漏洞挖掘方法,文献

[138]采用 LSTM 深度学习模型挖掘 PDF 文件对象的复杂结构,通过引入畸形数据对原样本域进行随机扰动,从而执行错误处理代码.

就目前而言,将深度学习应用于浏览器安全漏洞、文档类型漏洞和操作系统安全漏洞挖掘等方面的研究还在初期阶段.针对具体的应用场景设计针对性的漏洞挖掘模型,这方面的研究依然值得安全研究人员进行探索.

4 未来研究展望

通过梳理和归纳现有基于深度学习的软件漏洞挖掘的研究成果,一方面,本文以每种代码表征方式为出发点,分别在 2.1~2.5 节分类阐述和对比现有具有代表性的研究工作,并给出了详尽的讨论和观点(讨论 & 观点 1~5).另一方面,本文从不同的的应用场景出发,给出基于深度学习的漏洞挖掘模型在物联网、区块链智能合约和其他领域漏洞挖掘的研究进展,具体如 3.1~3.3 节表述,指出目前深度学习应用于具体应用领域的一些想法(讨论 & 观点 6~7).由于漏洞种类繁多、漏洞产生原理复杂和现有漏洞挖掘技术发展不完备等原因,实现自动化和智能化漏洞挖掘仍然还有很长的路要走.将深度学习应用于漏洞挖掘领域,虽已取得一定数量的代表性成果,但现阶段的发展仍未成熟.

本节基于 2~3 节对现有研究成果的一些讨论和观点(讨论 & 观点 1~7),尝试总结基于深度学习的漏洞挖掘领域现阶段面临的主要挑战,并在已有综述文献[139-147]之上,对未来的重点研究方向主要从 5 个方向进行展望.深度学习应用于漏洞挖掘领域研究的九大挑战和机遇如表 8 所示:

Table 8 Challenges and Opportunities in the Field of Vulnerability Mining based on Deep Learning

表 8 深度学习应用于漏洞挖掘领域研究面临的挑战和机遇

挑战	机遇
①漏洞数据集	建立一个公开规范且可以作为基准的数据集
②语义特征	构建新的数据表征方式
③语义信息爆炸	降维处理
④多种漏洞挖掘	多分类漏洞挖掘模型
⑤漏洞位置定位	细粒度的可解释模型
⑥深度学习算法	选择更符合漏洞检测应用场景的网络模型
⑦高误报和高漏报	将深度学习与动静态分析结合
⑧漏洞挖掘效率	中间代码表示
⑨跨项目漏洞挖掘	迁移学习用于跨语言、跨项目漏洞检测

4.1 漏洞数据集

将深度学习应用于自动化、智能化漏洞挖掘,首先面临的挑战是数据集的获取(挑战①),通过对现阶段该领域文献调研,发现现有各项研究在性能评估阶段都几乎依赖于各自收集和建立的数据集,尚未形成一个统一规范的数据集.同时,在深度学习训练模型过程中,需要足够准确的标记数据集,才能获得高度有效的训练结果.因此,为了辅助深度学习训练模型,必须构建一个公开规范且可以作为基准的数据集.

4.2 程序数据表征

基于深度学习的漏洞挖掘模型,根据提取特征的方式,可以分为基于序列表征、基于抽象语法树、基于图表征、基于文本表征和基于混合表征的漏洞挖掘模型.最大程度上完整保留程序数据的语法语义信息,抽象出代码蕴含的更深层信息,构建新的数据表征方式(挑战②),将有助于构建的神经网络训练模型拥有更好的检测性能.基于单特征表示的漏洞挖掘模型虽然在一定程度上能够抽象出程序数据的相关信息,但基于混合表征的漏洞挖掘模型在实际应用中具有更高的性能效果.因此,融合多种特征进行漏洞挖掘似乎是漏洞自动化挖掘的有效方案.针对语义模型中的特征爆炸问题(挑战③),可以采用特征降维方法提升模型的性能.同时,由于易受攻击的代码模式复杂,现有研究在特定类型的漏洞挖掘上取得了较好的性能,但实现多种漏洞挖掘(挑战④)仍待安全研究人员进一步进行探索.

4.3 深度学习模型

将深度学习应用于漏洞挖掘不需要安全专家手工定义漏洞特征,能够实现自动化和智能化漏洞特征提取.然而目前基于深度学习的漏洞挖掘面临的问题是如何将抽象出的程序数据表征转化为适合深度模型的向量表示形式.其次,程序数据具有丰富的层次结构和语法语义信息,尽管现有研究已经实现了从文件级别、函数级别到语句级别为粒度的漏洞检测,但仍然未提供与漏洞相关更全面的位置信息(挑战⑤).最后,不同的深度学习算法在同一数据集上会产生不同的性能效果,如何构建合适的深度学习模型(挑战⑥),挖掘更深层次的代码特征,实现细粒度的可解释漏洞挖掘模型,也需要进一步加以研究.

4.4 漏洞智能评估

传统的漏洞挖掘方法使用静态分析、动态分析等程序分析技术,虽取得一定的进展,却面临严重的高误报和高漏报等问题.同时,基于深度学习的漏洞

挖掘模型在一定程度上能够提升基于机器学习的漏洞挖掘的性能.因此,将深度学习与静、动态分析技术相结合进行漏洞挖掘能够提升模型的准确率,降低高误报和高漏报等问题(挑战⑦).面对复杂的深度学习模型以及多层次的代码表征方式,实现高效率的漏洞挖掘也是一大难题(挑战⑧).虽然现有的一部分工作采用 LLVM 等方法实现高效率的漏洞挖掘,但尚未找到有效的代码表征方式进行漏洞挖掘,该方面的研究也是未来的一个研究难点.

4.5 跨项目漏洞挖掘

跨项目安全漏洞挖掘(挑战⑨)是对不同项目的漏洞进行检测,需要将一个项目上构造的深度学习模型用于挖掘另外一个项目的漏洞.然而,由于不同项目的开发流程、应用领域、编程语言和开发人员的经验等差异,使得跨项目漏洞检测的难度加大.虽然有部分研究尝试采用迁移学习实现跨项目漏洞检测,但仅限于同种编程语言的不同项目之间,尚未实现跨语言的漏洞挖掘.由此看来,跨语言和跨项目的漏洞挖掘在未来依旧是一个值得探索的研究热点.

5 结束语

随着人工智能技术的不断发展,将深度学习应用于软件漏洞检测能够实现自动化和智能化漏洞挖掘,缓解了高误报率和高漏报率等问题.本文通过梳理和总结现有基于深度学习的漏洞挖掘最新研究,归纳其整体工作流程和技术路线,并从其中核心的深度表征方式为切入点,对现有研究成果进行分类阐述.同时也总结了不同应用场景中基于深度学习的漏洞挖掘方法的研究进展.最后,对该领域所面临的挑战和机遇进行展望.

通过总结现有研究工作,本文认为在未来的研究工作中:最大程度上抽象出漏洞特征的更深层信息,构建新的代码表征方式将有助于提升现有漏洞挖掘模型的性能.同时,采用迁移学习和注意力机制等对跨项目检测、漏洞位置定位等问题进一步分析,克服现有研究方法的局限性,将有助于提升漏洞挖掘的能力.

参 考 文 献

[1] National Institute of Standards and Technology. National vulnerability database [EB/OL]. [2021-05-11]. https://nvd.nist.gov/

[2] Wu Shizhong, Guo Tao, Dong Guowei, et al. Software vulnerability analyses: A road map [J]. Journal of Tsinghua University, 2012, 52(10): 1309-1319

[3] Ghaffarian S M, Shahriari H R. Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey [J]. ACM Computing Surveys, 2017, 50(4): 1-36

[4] Sabottke C, Suciu O, Dumitras T. Vulnerability disclosure in the age of social media: Exploiting Twitter for predicting real-world exploits [C] //Proc of the 24th USENIX Security Symp. Berkeley, CA: USENIX Association, 2015: 1041-1056

[5] National Information Security Standardization Technical Committee (SAC/TC260). GB/T28458—2012 Vulnerability identification and description specifications [S]. Beijing: China Standards Publishing House, 2012 (in Chinese)
(全国信息安全标准化技术委员会(SAC/TC260). GB/T28458—2012 信息安全技术安全漏洞标识与描述规范[S]. 北京: 中国标准出版社, 2012)

[6] Chess B, Gerschefske M. Rough Auditing Tool for Security [CP/OL]. [2019-03-18]. https://code.google.com/archive/p/rough-auditing-tool-for-security/

[7] Gao Debin, Reiter M K, Song D. Binhunt: Automatically finding semantic differences in binary programs [C] //Proc of the Int Conf on Information and Communications Security. Berlin: Springer, 2008: 238-255

[8] Ramos D A, Engler D R. Under-constrained symbolic execution: Correctness checking for real code [C] //Proc of the 24th USENIX Security Symp. Berkeley, CA: USENIX Association, 2015: 49-64

[9] King J C. Symbolic execution and program testing [J]. Communications of the ACM, 1976, 19(7): 385-394

[10] Sutton M, Greene A, Amini P. Fuzzing: Brute Force Vulnerability Discovery [M]. Piscataway, NJ: Pearson Education, 2007

[11] Wang Jinghan, Song Chengyu, Yin Heng. Reinforcement learning-based hierarchical seed scheduling for greybox fuzzing [C/OL]. [2021-08-30]. https://dx.doi.org/10.14722/ndss.2021.24486

[12] Xie Tao, Tillmann N, Halleux J, et al. Fitness-guided path exploration in dynamic symbolic execution [C] //Proc of the IEEE/IFIP Int Conf on Dependable Systems & Networks. Piscataway, NJ: IEEE, 2009: 359-368

[13] Poeplau S, Francillon A. SymQEMU: Compilation-based symbolic execution for binaries [C/OL]. [2021-08-30]. https://dx.doi.org/10.14722/ndss.2021.23118

[14] Humphries A, Subramanian K, Reiter M. TASE: Reducing latency of symbolic execution with transactional memory [C/OL]. [2021-08-30]. https://dx.doi.org/10.14722/ndss.2021.24327

[15] Newsome J, Song D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software [J]. Chinese Journal of Engineering Mathematics, 2005, 29(5): 720-724

[16] Ye Zhibin, Yan Bo. Survey of symbolic execution [J]. Computer Science, 2018, 45(6): 28-35 (in Chinese)
(叶志斌, 严波. 符号执行研究综述[J]. 计算机科学, 2018, 45(6): 28-35)

[17] Sun Hongyu, He Yuan, Wang Jice, et al. Application of artificial intelligence technology in the field of security vulnerability [J]. Journal on Communications, 2018, 39(8): 1-17 (in Chinese)
(孙鸿宇, 何远, 王基测, 等. 人工智能技术在安全漏洞领域的应用[J]. 通信学报, 2018, 39(8): 1-17)

[18] Dawoud A, Bugiel S. Bringing balance to the force: Dynamic analysis of the Android application framework [C/OL]. [2021-08-30]. https://dx.doi.org/10.14722/ndss.2021.23106

[19] Chen Qiuyu, Zhang Wei, Yu Jun, et al. Embedding complementary deep networks for image classification [C] //Proc of the IEEE/CVF Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2019: 9238-9247

[20] Wu Yue, Chen Yinpeng, Yuan Lu, et al. Rethinking classification and localization for object detection [C] //Proc of the IEEE/CVF Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2020: 10183-10192

[21] Ma Pingchuan, Wang Shuai, Liu Jin. Metamorphic testing and certified mitigation of fairness violations in NLP models [C] //Proc of the 29th Int Joint Conf on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 2020: 458-465

[22] MCCABE T J. A complexity measure [J]. IEEE Transactions on software Engineering, 1976, 2(4): 308-320

[23] Li Yun, Huang Chenlin, Wang Zhongfeng, et al. Survey of software vulnerability mining methods based on machine learning [J]. Journal of Software, 2020, 31(7): 2040-2061 (in Chinese)
(李韵, 黄辰林, 王中锋, 等. 基于机器学习的软件安全漏洞挖掘方法综述[J]. 软件学报, 2020, 31(7): 2040-2061)

[24] Shin Y, Meneely A, Williams L, et al. Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities [J]. IEEE Transactions on Software Engineering, 2011, 37(6): 772-787

[25] Shin Y, Williams L. An initial study on the use of execution complexity metrics as indicators of software vulnerabilities [C] //Proc of the 7th Int Workshop on Software Engineering for Secure Systems. New York: ACM, 2011: 1-7

[26] Chowdhury I, Zulkernine M. Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities [J]. Journal of Systems Architecture, 2011, 57(3): 294-313

[27] Younis A, Malaiya Y K, Anderson C, et al. To fear or not to fear that is the question: Code characteristics of a vulnerable function with an existing exploit [C] //Proc of the Conf on Data & Applications Security & Privacy. New York: ACM, 2016: 97-104

[28] Bozorgi M, Saul L K, Savage S, et al. Beyond heuristics: Learning to classify vulnerabilities and predict exploits [C] //Proc of the 16th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: ACM, 2010: 105-114

[29] Perl H, Dechand S, Smith M, et al. VCCFinder: Finding potential vulnerabilities in open-source projects to assist code audits [C] //Proc of the ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2015: 426-437

[30] Shar L K, Tan H. Predicting SQL injection and cross site scripting vulnerabilities through mining input sanitization patterns [J]. Information and Software Technology, 2013, 55(10): 1767-1780

[31] Li Zhenmin, Zhou Yuanyuan. PR-miner: Automatically extracting implicit programming rules and detecting violations in large software code [J]. ACM SIGSOFT Software Engineering Notes, 2005, 30(5): 306-315

[32] Yamaguchi F, Wressnegger C, Gascon H, et al. Chucky: Exposing missing checks in source code for vulnerability discovery [C] //Proc of 2013 ACM SIGSAC Conf on Computer & Communications Security. New York: ACM, 2013: 499-510

[33] Padmanabhuni B M, Tan H. Buffer overflow vulnerability prediction from x86 executables using static analysis and machine learning [C] //Proc of 2015 IEEE 39th Annual Computer Software and Applications Conf. Piscataway, NJ: IEEE, 2015: 450-459

[34] Meng Qingkun, Wen Shameng, Feng Chao, et al. Predicting integer overflow through static integer operation attributes [C] //Proc of 2016 5th Int Conf on Computer Science and Network Technology (ICCSNT). Piscataway, NJ: IEEE, 2016: 177-181

[35] Shar L K, Tan H, Briand L C. Mining SQL injection and cross site scripting vulnerabilities using hybrid program analysis [C] //Proc of 2013 Int Conf on Software Engineering (ICSE). Piscataway, NJ: IEEE, 2013: 642-651

[36] Shar L K, Briand L C, Tan H. Web application vulnerability prediction using hybrid program analysis and machine learning [J]. IEEE Transactions on Dependable and Secure Computing, 2015, 12(6): 688-707

[37] Jang J, Agrawal A, Brumley D. ReDeBug: Finding unpatched code clones in entire OS distributions [C] //Proc of 2012 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2012: 48-62

[38] Li Zhen, Zou Deqing, Xu Shouhai, et al. VulPecker: An automated vulnerability detection system based on code similarity analysis [C] //Proc of the 32nd Annual on Computer Security Applications Conf (ACSAC). New York: ACM, 2016: 201-213

[39] Kim S, Woo S, Lee H, et al. VUDDY: A scalable approach for vulnerable code clone discovery [C] //Proc of 2017 IEEE Symp on Security and Privacy (SP). Piscataway, NJ: IEEE, 2017: 595-614

[40] Yamaguchi F, Lindner F, Rieck K. Vulnerability extrapolation: Assisted discovery of vulnerabilities using machine learning [C] //Proc of the 5th USENIX Workshop on Offensive Technologies. Berkeley, CA: USENIX Association, 2011: 118-127

[41] Scandariato R, Walden J, Hovsepyan A, et al. Predicting vulnerable software components via text mining [J]. IEEE Transactions on Software Engineering, 2014, 40(10): 993-1006

[42] Chemis B, Verma R. Machine learning methods for software vulnerability detection [C] //Proc of the 4th ACM Int Workshop on Security and Privacy Analytics(IWSPA). New York: ACM, 2018: 31-39

[43] Wijayasekara D, Manic M, Mcqueen M. Information gain based dimensionality selection for classifying text documents [C] //Proc of 2013 IEEE Congress on Evolutionary Computation (CEC). Piscataway, NJ: IEEE, 2013: 440-445

[44] Wijayasekara D, Manic M, Mcqueen M. Vulnerability identification and classification via text mining bug databases [C] //Proc of the 40th Annual Conf of the IEEE Industrial Electronics Society. Piscataway, NJ: IEEE, 2014: 3612-3618

[45] Hovsepyan A, Scandariato R, Joosen W, et al. Software vulnerability prediction using text analysis techniques [C] //Proc of the 4th Int Workshop on Security Measurements and Metrics. New York: ACM, 2012: 7-10

[46] Yamaguchi F, Lottmann M, Rieck K. Generalized vulnerability extrapolation using abstract syntax trees [C] //Proc of the 28th Annual Computer Security Applications Conf(ACSAC). New York: ACM, 2012: 359-368

[47] Medeiros I, Neves N F, Correia M. Automatic detection and correction of Web application vulnerabilities using data mining to predict false positives [C] //Proc of the 23rd Int Conf on World Wide Web. New York: ACM, 2014: 63-74

[48] Anbiya D R, Purwarianti A, Asnar Y. Vulnerability detection in php Web application using lexical analysis approach with machine learning [C] //Proc of the 5th Int Conf on Data and Software Engineering (ICoDSE). Piscataway, NJ: IEEE, 2018: 170-175

[49] Meng Qingkun, Wen Shameng, Feng Chao, et al. Predicting buffer overflow using semi-supervised learning [C] //Proc of the 9th Int Congress on Image and Signal Processing. Piscataway, NJ: IEEE, 2016: 1959-1963

[50] Yamaguchi F, Golde N, Arp D, et al. Modeling and discovering vulnerabilities with code property graphs [C] //Proc of 2014 IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2014: 590-604

[51] Yamaguchi F, Maier A, Gascon H, et al. Automatic inference of search patterns for taint-style vulnerabilities [C] //Proc of the IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2015: 797-812

[52] Nguyen H, Tran L. Predicting vulnerable software components with dependency graphs [C] //Proc of the 6th Int Workshop on Security Measurements and Metrics. New York: ACM, 2010: 1-8

[53] Qian Feng, Zhou Rundong, Xu Chengcheng, et al. Scalable graph-based bug search for firmware images [C] //Proc of the ACM SIGSAC Conf on Computer & Communications Security(CCS). New York: ACM, 2016: 480-491

[54] Harer J A, Kim L Y, Russell R L, et al. Automated software vulnerability detection with machine learning [C/OL]. [2018-10-02]. https://arxiv.org/abs/1803.04497

[55] Mitra S, Avra L J, McCluskey E J. Scan synthesis for one-hot signals [C] //Proc of the Int Test Conf. Piscataway, NJ: IEEE, 1997: 714-722

[56] Mikolov T, Sutskever I, Chen Kai, et al. Distributed representations of words and phrases and their compositionality [C] //Proc of the 26th Int Conf on Neural Information Processing Systems. New York: ACM, 2013: 3111-3119

[57] Moghadasi M N, Zhang Yu. Sent2Vec: A new sentence embedding representation with sentimental semantic [C] //Proc of 2020 IEEE Int Conf on Big Data. Piscataway, NJ: IEEE, 2020: 4672-4680

[58] Li Xin, Wang Lu, Xin Yang, et al. Automated software vulnerability detection based on hybrid neural network [J]. Applied Sciences, 2021, 11(7): 3201-3216

[59] Li Zhen, Zou Deqing, Xu Shouhai, et al. VulDeePecker: A deep learning-based system for vulnerability detection [C] //Proc of the 25th Annual Network and Distributed System Security Symp(NDSS). San Diego, CA: Internet Society, 2018: 1-15

[60] Zou Deqing, Wang Sujuan, XuShouhai, et al. μVulDeePecker: A deep learning-based system for multiclass vulnerability detection [J]. IEEE Transactions on Dependable and Secure Computing, 2019, 18(5): 2224-2236

[61] Nguyen T, Le T, Nguyen K, et al. Deep Cost-Sensitive Kernel Machine for Binary Software Vulnerability Detection [M]. Berlin: Springer, 2020: 164-177

[62] Wu Fang, Wang Jigang, Liu Jiqiang, et al. Vulnerability detection with deep learning [C] //Proc of the 3rd IEEE Int Conf on Computer and Communications (ICCC). Piscataway, NJ: IEEE, 2017: 1298-1302

[63] Chen Haipeng, Liu Jing, Liu Rui, et al. VASE: A twitter-based vulnerability analysis and score engine [C] //Proc of the IEEE Int Conf on Data Mining(ICDM). Piscataway, NJ: IEEE, 2019: 976-981

[64] Pang Yulei, Xue Xiaozhen, Wang Huaying. Predicting vulnerable software components through deep neural network [C] //Proc of the 12th Int Conf on Advanced Computational Intelligence (ICACI). New York: ACM, 2017: 6-10

[65] Pechenkin A, Demidov R. Applying deep learning and vector representation for software vulnerabilities detection [C] //Proc of the 11th Int Conf on Security of Information and Networks. New York: ACM, 2018: 83-88

[66] Shen Yun, Mariconti E, Vervier V A, et al. Tiresias: Predicting security events through deep learning [C] //Proc of 2018 ACM SIGSAC Conf on Computer and Communications Security(CCS). New York: ACM, 2019: 592-605

[67] Fang Yong, Han Shengjun, Huang Cheng, et al. TAP: A static analysis model for PHP vulnerabilities based on token and deep learning technology [J]. PLoS ONE, 2019, 14(11): 1-19

[68] Aghaei E, Al-shaer E. ThreatZoom: Neural network for automated vulnerability mitigation [C] // Proc of the 6th Annual Symp on Hot Topics in the Science of Security (HotSoS). New York: ACM, 2019: 102-104

[69] Jabeen G, Luo Ping, Akram J, et al. An integrated software vulnerability discovery model based on artificial neural network [C] //Proc of the 31st Int Conf on Software Engineering and Knowledge Engineering. Pittsburgh: KSI, 2019: 349-458

[70] Niu Weina, Zhang Xiaosong, Du Xiaojiang, et al. A deep learning based static taint analysis approach for IoT software vulnerability location [J]. Measurement, 2019, 152(3): 107-139

[71] Wu Shuang, Wang Congyi, Zeng Jianping, et al. Vulnerability time series prediction based on multivariable LSTM [C] // Proc of the IEEE 14th Intl Conf on Anti-counterfeiting, Security, and Identification (ASID). Piscataway, NJ: IEEE, 2020: 185-190

[72] Nguyen V, Le T, Vel O D, et al. Dual-component deep domain adaptation: A new approach for cross project software vulnerability detection [C] //Proc of the Pacific-Asia Conf on Knowledge Discovery and Data Mining. Berlin: Springer, 2020: 699-711

[73] Pang Yulei, Xue Xiaozhen, Namin A S. Predicting vulnerable software components through *n*-gram analysis and statistical feature selection [C] //Proc of the 14th Int Conf on Machine Learning and Applications (ICMLA). Piscataway, NJ: IEEE, 2015: 543-548

[74] Feng Hantao, Fu Xiaotong, Sun Hongyu, et al. Efficient vulnerability detection based on abstract syntax tree and deep learning [C] //Proc of the IEEE Conf on Computer Communications Workshops (INFOCOM WKSHPS). Piscataway, NJ: IEEE, 2020: 722-727

[75] Zhang Jian, Wang Xu, Zhang Hongyu, et al. A novel neural source code representation based on abstract syntax tree [C] //Proc of 2019 IEEE/ACM 41st Int Conf on Software Engineering (ICSE). New York: ACM, 2019: 783-794

[76] Cao Defu, Huang Jing, Zhang Xuanyu, et al. FTCLNet: Convolutional LSTM with Fourier transform for vulnerability detection [C] //Proc of the 19th IEEE Int Conf on Trust, Security and Privacy in Computing and Communications (TrustCom). Piscataway, NJ: IEEE, 2020: 539-546

[77] Song Wang, Liu Taiyue, Tan Lin. Automatically learning semantic features for defect prediction [C] //Proc of the 38th Int Conf on Software Engineering. Piscataway, NJ: IEEE, 2016: 297-308

[78] LinGuanjun, Zhang Jun, Luo Wei, et al. Cross-project transfer representation learning for vulnerable function discovery [J]. IEEE Transactions on Industrial Informatics, 2018, 14(7): 3289-3297

[79] Mao Yi, Li Yun, Sun Jiatai, et al. Explainable software vulnerability detection based on attention-based bidirectional recurrent neural networks [C] //Proc of 2020 IEEE Int Conf on Big Data. Piscataway, NJ: IEEE, 2020: 4651-4656

[80] Mou Lili, Li Ge, Zhang Lu, et al. Convolutional neural networks over tree structures for programming language processing [C] //Proc of the 13th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2016: 1287-1293

[81] Dam H, Tran T, Pham T, et al. Automatic feature learning for vulnerability prediction [J/OL]. [2017-10-08]. https://arxiv.org/abs/1708.02368

[82] Lin Guanjun, Zhang Jun, Luo Wei, et al. POSTER: Vulnerability discovery with function representation learning from unlabeled projects [C] //Proc of 2017 ACM SIGSAC Conf on Computer and Communications Security(CCS). New York: ACM, 2017: 2539-2541

[83] Dam H K, Pham T, Ng S W, et al. A deep tree-based model for software defect prediction [J/OL]. [2018-02-03]. https://arxiv.org/abs/1802.00921

[84] Liu Shigang, Lin Guanjun, Han Qinglong, et al. DeepBalance: Deep-learning and fuzzy oversampling for vulnerability detection [J]. IEEE Transactions on Fuzzy Systems, 2020, 28(7): 1329-1343

[85] Buch L, Andrzejak A. Learning-based recursive aggregation of abstract syntax trees for code clone detection [C] //Proc of 2019 IEEE 26th Int Conf on Software Analysis, Evolution and Reengineering (SANER). Piscataway, NJ: IEEE, 2019: 95-104

[86] Liu Shigang, Lin Guanjun, Qu Lizhen, et al. CD-VulD: Cross-domain vulnerability discovery based on deep domain adaptation [J/OL]. IEEE Transactions on Dependable and Secure Computing, 2020 [2021-09-06]. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9054952

[87] He Yuan, Sun Hongyu, Feng Hantao. UA-Miner: Deep learning systems for expose unprotected API vulnerability in source code [C] //Proc of the 12th Int Conf on Advanced Computational Intelligence (ICACI). Piscataway, NJ: IEEE, 2020: 378-384

[88] Li Xin, Wang Lu, Xin Yang, et al. Automated vulnerability detection in source code using minimum intermediate representation learning [J]. Applied Sciences, 2020, 10(5): 1692

[89] Tian Junfeng, Xing Wenjing, Li Zhen. BVDetector: A program slice-based binary code vulnerability intelligent detection system [J/OL]. Information and Software Technology, 2020 [2020-02-18]. https://doi.org/10.1016/j.infsof.2020.106289

[90] Allamanis M, Brockschmidt M, Khademi M. Learning to represent programs with graphs [C/OL] //Proc of the Int Conf on Learning Presentations(ICLR). 2017 [2017-11-01]. https://arxiv.org/abs/1711.00740v3

[91] Zeng Jingxiang, Nie Xiaofan, Chen Liwei, et al. An efficient vulnerability extrapolation using similarity of graph kernel of PDGs [C] //Proc of the 19th Int Conf on Trust, Security and Privacy in Computing and Communications (TrustCom). Piscataway, NJ: IEEE, 2020, 1664-1671

[92] Guo Ning, Li Xiaoyong, Yin Hui, et al. VulHunter: An automated vulnerability detection system based on deep learning and bytecode[C]//Proc of the Int Conf on Information and Communication. Berlin: Springer, 2019: 199-218

[93] Chen Peng, Liu Jianzhong, Chen Hao. Matryoshka: Fuzzing deeply nested branches[C]//Proc of 2019 ACM SIGSAC Conf on Computer and Communications Security (CCS). New York: ACM, 2019: 499-513

[94] Duan Xu, Wu Jingzheng, Ji Shouling, et al. VulSniper: Focus your attention to shoot fine-grained vulnerabilities[C]//Proc of the 28th Int Joint Conf on Artificial Intelligence (IJCAI). San Francisco, CA: Morgan Kaufmann, 2019: 4665-4671

[95] Wang Xiaomeng, Zhang Tao, Wu Runpu, et al. CPGVA: Code property graph based vulnerability analysis by deep learning[C]//Proc of the 10th Int Conf on Advanced Infocomm Technology (ICAIT). Piscataway, NJ: IEEE, 2018: 184-188

[96] Zhao Gang, Huang J. DeepSim: Deep learning code functional similarity[C]//Proc of the 26th ACM Joint Meeting on European Software Engineering Conf and Symp on the Foundations of Software Engineering (ESEC/FSE). New York: ACM, 2018: 141-151

[97] Peng Hao, Mou Lili, Li Ge, et al. Building program vector representations for deep learning[C]//Proc of the 8th Int Conf on Knowledge Science, Engineering and Management. Berlin: Springer, 2015: 547-553

[98] Sun Mingxin, Wang Wenjie, Feng Hantao, et al. Identify vulnerability fix commits automatically using hierarchical attention network[J]. Security and Safety, 2018, 7(23): 17-31

[99] Wang Lu, Li Xin, Wang Ruineng, et al. PreNNsem: A heterogeneous ensemble learning framework for vulnerability detection in software[J]. Applied Sciences, 2020, 10(22): 7954-7969

[100] Ziems N, Wu Shaoen. Security vulnerability detection using deep learning natural language processing[C/OL]//Proc of 2021 IEEE Int Conf on Computer Communications (IEEE INFOCOM). [2021-05-06]. https://arxiv.org/abs/2105.02388

[101] Choi M J, Jeong S, Oh H, et al. End-to-end prediction of buffer overruns from raw source code via neural memory networks[C]//Proc of the 26th Int Joint Conf on Artificial Intelligence (IJCAI). San Francisco, CA: Morgan Kaufmann, 2017: 1546-1553

[102] Fidalgo A, Medeiros I, Antunes P, et al. Towards a deep learning model for vulnerability detection on Web application variants[C]//Proc of 2020 IEEE Int Conf on Software Testing, Verification and Validation Workshops. Piscataway, NJ: IEEE, 2020: 465-476

[103] Russell R L, Kim L, Hamilton L, et al. Automated vulnerability detection in source code using deep representation learning[C]//Proc of the 17th IEEE Int Conf on Machine Learning and Applications (ICMLA). Piscataway, NJ: IEEE, 2018: 757-762

[104] Nembhard F, Carvalho M, Eskridge T. Extracting knowledge from open source projects to improve program security[C]//Proc of the IEEE South East Conf. Piscataway, NJ: IEEE, 2018: 631-637

[105] Sestili C D, Snavely W S, Vanhoudnos N M. Towards security defect prediction with AI[C/OL]. [2018-09-12]. https://arxiv.org/abs/1808.09897

[106] Le T, Nguyen T, Le T, et al. Maximal divergence sequential auto-encoder for binary software vulnerability detection[C/OL]. [2019-02-26]. https://openreview.net/forum?id=ByloIiCqYQ

[107] Lee Y J, Kwon H, Choi S, et al. Instruction2vec: Efficient preprocessor of assembly code to detect software weakness with CNN[J]. Applied Sciences, 2019, 9(19): 4086-4101

[108] Dong Feng, Wang Junfeng, Li Qi, et al. Defect prediction in Android binary executables using deep neural network[J]. Wireless Personal Communications, 2018, 102(3): 2261-2285

[109] Zhou Yaqin, Liu Shangqing, Siow J, et al. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks[C]//Proc of the Advances in Neural Information Processing Systems (NeurLPS 2019). San Francisco, CA: Morgan Kaufmann, 2019: 10197-10207

[110] LiZhen, Zou Deqing, Xu Shouhai, et al. SySeVR: A framework for using deep learning to detect software vulnerabilities[J/OL]. [2021-01-12]. https://arxiv.org/abs/1807.06756

[111] Li Zhen, Zou Deqing, Xu Shouhai, et al. VulDeeLocator: A deep learning-based fine-grained vulnerability detector[J/OL]. IEEE Transactions on Dependable and Secure Computing, 2021 [2021-05-01]. https://arxiv.org/abs/2001.02350

[112] Wu Yuelong, Lu Jintian, Zhang Yunyi, et al. Vulnerability detection in C/C++ source code with graph representation learning[C]//Proc of 2021 IEEE 11th Annual Computing and Communication Workshop and Conf (CCWC). Piscataway, NJ: IEEE, 2021: 1519-1524

[113] Jeon S, Kim H K. AutoVAS: An automated vulnerability analysis system with a deep learning approach[J/OL]. Computers & Security, 2021 [2021-04-19]. https://doi.org/10.1016/j.cose.2021.102308

[114] Cao Sicong, Sun xiaobing, Bo lili, et al. BGNN4VD: Constructing bidirectional graph neural-network for vulnerability detection[J/OL]. Information and Software Technology, 2021 [2021-03-15]. https://doi.org/10.1016/j.infsof.2021.106576

[115] Wang Huanting, Ye Guixin, Tang Zhanyong, et al. Combining graph-based learning with automated data collection for code vulnerability detection[J/OL]. IEEE Transactions on Information Forensics and Security, 2021: 1943-1958. [2021-03-15]. https://doi.org/10.1109/TIFS.2020.3044773

[116] Cui Lei, Hao Zhiyun, Jiao Yang, et al. VulDetector: Detecting vulnerabilities using weighted feature graph comparison [J/OL]. IEEE Transactions on Information Forensics and Security, 2021: 2004-2017. [2021-03-15]. https://doi.org/10.1109/TIFS.2020.3047756

[117] Bowman B, Huang H H. VGRAPH: A robust vulnerable code clone detection system using code property triplets [C] //Proc of 2020 IEEE European Symp on Security and Privacy (EuroS&P). Piscataway, NJ: IEEE, 2020: 53-69

[118] Duan Xu, Wu Jingzheng, Luo Tianyue, et al. Vulnerability mining method based on code property graph and attention BiLSTM [J]. Journal of Software, 2020, 31(11): 88-104 (in Chinese)
(段旭, 吴敬征, 罗天悦, 等. 基于代码属性图及注意力双向LSTM的漏洞挖掘方法[J]. 软件学报, 2020, 31(11): 88-104)

[119] Feng Qi, Feng Chengdong, Hong Weijiang. Graph neural network-based vulnerability predication [C] //Proc of 2020 IEEE Int Conf on Software Maintenance and Evolution (ICSME). Piscataway, NJ: IEEE, 2020: 800-801

[120] Zuo Fei, Li Xiaopeng, Youn P, et al. Neural machine translation inspired binary code similarity comparison beyond function pairs [C/OL] //Proc of the 25th Annual Network and Distributed System Security Symp (NDSS). San Diego, CA: Internet Society, 2019 [2019-02-24]. https://dx.doi.org/10.14722/ndss.2019.23492

[121] White M, Tufano M, Martinez M, et al. Sorting and transforming program repair ingredients via deep learning code similarities [C] //Proc of 2019 IEEE 26th Int Conf on Software Analysis, Evolution and Reengineering (SANER). Piscataway, NJ: IEEE, 2019: 479-490

[122] Lin Guanjun, Zhang Jun, Luo Wei, et al. Software vulnerability discovery via learning multi-domain knowledge bases [J]. IEEE Transactions on Dependable and Secure Computing, 2019, 18(5): 2469-2485

[123] White M, Tufano M, Vendome C, et al. Deep learning code fragments for code clone detection [C] //Proc of the 31st IEEE/ACM Int Conf on Automated Software Engineering. New York: ACM, 2016: 87-98

[124] Xu Xiaojun, Liu Chang, Qian Feng, et al. Neural network-based graph embedding for cross-platform binary code similarity detection [C] //Proc of the ACM SIGSAC Conf on Computer and Communications Security. New York: ACM, 2017: 363-376

[125] Gao Jian, Yang Xin, Fu Ying, et al. Vulseeker: A semantic learning based vulnerability seeker forcross-platform binary [C] //Proc of the 33rd ACM/IEEE Int Conf on Automated Software Engineering. New York: ACM, 2018: 896-899

[126] Gao Jian, Yang Xin, Fu Ying, et al. Vulseeker-pro: Enhanced semantic learning based binary vulnerability seeker with emulation [C] //Proc of the 26th ACM Joint Meeting on European Software Engineering Conf and Symp on the Foundations of Software Engineering. New York: ACM, 2018: 803-808

[127] Baldoni R, Luna G A, Massarelli L, et al. Unsupervised features extraction for binary similarity using graph embedding neural network [J/OL]. [2018-11-13]. https://arxiv.org/abs/1810.09683

[128] Yu Zeping, Cao Rui, Tang Qiyi, et al. Order Matters: Semantic-aware neural networks for binary code similarity detection [C] //Proc of 2020 AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2020, 34(1): 1145-1152

[129] Liu Bingchang, Huo Wei, Zhang Chao, et al. αDiff: Cross-version binary code similarity detection with DNN [C] //Proc of the 33rd ACM/IEEE Int Conf on Automated Software Engineering. New York: ACM, 2018: 667-678

[130] Tann W, Han Xingjie, Gupta S, et al. Towards safer smart contracts: A sequence learning approach to detecting vulnerabilities [C/OL]. [2019-08-07]. https://arxiv. org/abs/1811.06632

[131] Huang T H. Hunting the ethereum smart contract: Color-inspired inspection of potential attacks [C/OL]. [2018-07-05]. https://arxiv.org/pdf/1807.01868

[132] Zhuang Yuan, Liu Zhenguang, Qian Peng, et al. Smart contract vulnerability detection using graph neural networks [C] //Proc of the 29th Int Joint Conf on Artificial Intelligence and 17th Pacific Rim Int Conf on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 2020: 3255-3262

[133] HeJianxuan, Balunovic M, Ambroladze N, et al. Learning to fuzz from symbolic execution with application to smart contracts [C] //Proc of 2019 ACM SIGSAC Conf on Computer and Communications Security(CCS). New York: ACM, 2019: 531-548

[134] Lutz O, Chen Huili, Fereidooni H, et al. ESCORT: Ethereum smart COntRacTs vulnerability detectionusing deep neural network and transfer learning [C/OL]. [2021-03-23]. https://arxiv.org/abs/2103.12607

[135] Ashizawa N, Yanai N, Cruz J P, et al. Eth2Vec: Learning contract-wide code representations for vulnerability detection on ethereum smart contracts [C/OL]. [2021-01-08]. https://arxiv.org/abs/2101.02377

[136] Grieco G, Grinblat G L, Uzal L, et al. Toward large-scale vulnerability discovery using machine learning [C] //Proc of the Sixth ACM Conf on Data and Application Security and Privacy. New York: ACM, 2016: 85-96

[137] Lv Chenyang, Ji Shouling, Li Yuwei, et al. Smartseed: Smart seed generation for efficient fuzzing [J/OL]. [2019-08-03]. https://arxiv.org/abs/1807.02606

[138] Godefroid P, Peleg H, Singh R. Learn&-fuzz: Machine learning for input fuzzing [C] //Proc of the 32nd IEEE/ACM Int Conf on Automated Software Engineering. Piscataway, NJ: IEEE, 2017: 50-59

[139] Nguyen H N, Teerakanok S, Inomata A, et al. The comparison of word embedding techniques in RNNs for vulnerability detection [C] //Proc of the 7th Int Conf on Information Systems Security and Privacy(ICISSP). Seubal, Portugal: STP, 2021: 109-120

[140] Tang Gaigai, Meng Lianxiao, Wang Huiqiang, et al. A comparative study of neural network techniques for automatic software vulnerability detection [C] //Proc of 2020 Int Sympon Theoretical Aspects of Software Engineering (TASE). Piscataway, NJ: IEEE, 2020: 1-8

[141] Guan Zhibin, Wang Xiaomeng, Xin Wei, et al. A survey on deep learning-based source code defect analysis [C] //Proc of 2020 5th Int Conf on Computer and Communication Systems (ICCCS). Piscataway, NJ: IEEE, 2020: 167-171

[142] Chakraborty S, Krishna R, Ding Y, et al. Deep learning based vulnerability detection: Are we there yet? [J/OL]. [2020-09-03]. https://arxiv.org/abs/2009.07235

[143] Lin Guanjun, Xiao Wei, Zhang Jun, et al. Deep learning-based vulnerable function detection: A benchmark [C] // Proc of the Int Conf on Information and Communications Security. Berlin: Springer, 2019: 219-232

[144] Lin Guanjun, Wen Sheng, Han Qinglong, et al. Software vulnerability detection using deep neural networks: A survey [J]. Proceedings of the IEEE, 2020, 108(10): 1825-1848

[145] LiZhen, Zou Deqing, Wang Zeli, et al. Survey on static software vulnerability detection for source code [J]. Chinese Journal of Network and Information Security, 2019, 5(1): 1-14 (in Chinese)
(李珍, 邹德清, 王泽丽, 等. 面向源代码的软件漏洞静态检测综述[J]. 网络与信息安全学报, 2019, 5(1): 1-14)

[146] Ban Xinbo, Liu Shigang, Chen Chao, et al. A performance evaluation of deep-learnt features for software vulnerability detection [J]. Concurrency and Computation: Practice and Experience, 2019, 31(19): 5103-5114

[147] Choi Y H, Liu Peng, Shang Zitong, et al. Using deep learning to solve computer security challenges: A survey [J]. Cybersecurity, 2020, 3(1): 15-63

Gu Mianxue, born in 1994. PhD candidate. His main research interests include network and information system security.
顾绵雪,1994 年生.博士研究生.主要研究方向为网络与信息系统安全.

Sun Hongyu, born in 1993. PhD candidate. His main research interests include information security and machine learning.
孙鸿宇,1993 年生.博士研究生.主要研究方向为信息安全和机器学习.

Han Dan, born in 1998. Master candidate. Her main research interests include network and information system security.
韩 丹,1998 年生.硕士研究生.主要研究方向为网络与信息系统安全.

Yang Su, born in 1993. PhD candidate. His main research interests include information security and vulnerability mining.
杨 粟,1993 年生.博士研究生.主要研究方向为信息安全和漏洞挖掘.

Cao Wanying, born in 2000. PhD candidate. Her main research interests include network and information system security.
曹婉莹,2000 年生.博士研究生.主要研究方向为网络与信息系统安全.

Guo Zhen, born in 1981. PhD. Master supervisor. Her main research interests include information system security and data privacy preservation.
郭 祯,1981 年生.博士,硕士生导师.主要研究方向为信息系统安全和数据隐私保护.

Cao Chunjie, born in 1977. PhD, professor, PhD supervisor. His main research interests include communication and network security, network confrontation.
曹春杰,1977 年生.博士,教授,博士生导师.主要研究方向为通信与网络安全、网络对抗.

Wang Wenjie, born in 1964. PhD, associate professor of University of Chinese Academy of Sciences. His main research interests include information security and intelligent information processing.
王文杰,1964 年生.博士,副教授.主要研究方向为信息安全和智能信息处理.

Zhang Yuqing, born in 1966. PhD, professor, PhD supervisor. His main research interest is information security.
张玉清,1966 年生.博士,教授,博士生导师.主要研究方向为信息安全.