•人工智能安全•

DOI:10.15961/j.jsuese.202100880

本刊网刊

基于语义感知图神经网络的智能合约字节码漏洞检测方法

赵　波[1], 上官晨晗[1*], 彭小燕[2], 安　扬[3], 童俊成[1], 袁安琪[1]

(1.武汉大学 国家网络安全学院, 湖北 武汉 430072; 2.上海航天电子通讯设备研究所, 上海 201109;
3.武汉大学 计算机学院, 湖北 武汉 430072)

摘　要:针对传统智能合约漏洞检测方法检测精度较低、误报率较高, 以及基于神经网络的方法对字节码级智能合约特征挖掘不足的问题, 提出了一种基于语义感知图神经网络的智能合约字节码漏洞检测方法。首先, 以智能合约字节码划分基本块作为节点, 并从字节码中提取基本块间的调用关系作为边, 以此生成控制流图(control flow graph, CFG), 传入图卷积神经网络(graph convolutional network, GCN)中进行训练得到图节点的特征表示; 其次, 对合约字节码指令序列进行分词, 再转化为词向量嵌入到低维空间, 传入长短期记忆(long short-term memory, LSTM)网络进行训练, 得到字节码语义信息的向量表示; 最后, 将生成的节点特征和语义特征进行拼接后传入全连接层进行降维, 结合语义信息和节点特征对智能合约进行漏洞检测。利用公开数据集中的真实智能合约进行训练和测试, 在通过传统方法和人工标签的两类漏洞分类数据集中进行验证。使用本文提出的方法与3种传统智能合约漏洞检测工具及1种基于神经网络的智能合约漏洞检测方法进行对比。实验结果表明本文提出的基于语义感知图神经网络智能合约字节码漏洞检测方法在各类指标上均有较大提升, 能够检测出其余4种方法未检测出的具有漏洞的合约, 说明在图神经网络中加入字节码语义信息能够有效提升检测精度, 降低误报率。

关键词:智能合约字节码; 图卷积神经网络; 语义感知; 漏洞检测

中图分类号:TP391.7　　**文献标志码**:A　　**文章编号**:2096-3246(2022)02-0049-07

Semantic-aware Graph Neural Network for Smart Contract Bytecode Vulnerability Detection

ZHAO Bo[1], *SHANGGUAN Chenhan*[1*], *PENG Xiaoyan*[2], *AN Yang*[3], *TONG Juncheng*[1], *YUAN Anqi*[1]

(1.School of Cyber Sci. and Eng., Wuhan Univ., Wuhan 430072, China;
2.Shanghai Aerospace Electronic Communication Equipment Inst., Shanghai 201109, China;
3.School of Computer Sci., Wuhan Univ., Wuhan 430072, China)

Abstract: In order to solve the problems of low detection accuracy and high false positive rate of traditional smart contract vulnerability detection methods and less consideration of bytecode level smart contract features in neural networks, a smart contract bytecode vulnerability detection method based on semantic perception graph neural network was proposed. First, in order to generate the control flow graph, the basic blocks divided by the smart contract bytecode were used as the nodes, and the call relationship between the basic blocks was extracted from the bytecode as the edges. Then, control flow graph is transmitted into the graph convolutional network for training to obtain the feature representation of the graph nodes; Afterwards, the contract bytecode instruction sequence is segmented, transformed into a word vector, embedded into a low-dimensional space and transmitted to a long short-term memory network for training. Then, the vector representation of bytecode semantic information

收稿日期:2021 – 08 – 31

基金项目:湖北省重点研发计划项目(2020BAB101; 2020BAA003); 上海航天科技创新基金项目(SAST2019–098); 国家自然科学基金联合基金项目(U1936122)

作者简介:赵　波(1972—), 男, 教授, 博士生导师, 博士. 研究方向: 信息系统安全; 可信计算; 嵌入式安全; 区块链安全; 人工智能及大数据安全隐私保护. E-mail: zhaobo@whu.edu.cn

* **通信作者**:上官晨晗, E-mail: mr_sinco@whu.edu.cn

网络出版时间:2022 – 03 – 10 13 : 44 : 52　　**网络出版地址**:https://kns.cnki.net/kcms/detail/51.1773.TB.20220309.1005.004.html

was obtained. Finally, the generated node features and semantic features were spliced and transmitted to the full connection layer for dimensionality reduction. Combined with semantic information and node features, the vulnerability detection was carried out for smart contracts. The real smart contracts in public dataset were used for training and testing, and verified in two types of vulnerability classification datasets through traditional methods and artificial tags. The method proposed in this paper was compared with three traditional smart contract vulnerability detection tools and one smart contract vulnerability detection method based on neural network. The experimental results showed that the proposed network greatly improves the performance of network in terms of various indicators, and detects the contracts with vulnerabilities which are not detected by the other four methods. It shows that adding the bytecode semantic information to graph neural network can effectively improve the detection accuracy and reduce the false alarm rate.

Key words: smart contract bytecode; GCN; semantic-aware; vulnerability detection

智能合约最早由尼克·萨博在20世纪90年代提出[1]，能够通过技术手段来强制保证合同条款在条件满足后被自动执行,无需可信第三方的监督。而区块链技术[2]的出现恰好为智能合约提供了去中心化和不可篡改的可信运行平台。

随着区块链在学术界和工业界的广泛研究与应用，智能合约存在的安全问题开始受到各方关注。由于智能合约在区块链上通常用于管理加密数字资产，因此更容易成为攻击者的目标。一旦智能合约存在漏洞遭受攻击，就可能造成重大经济损失。2016年，The DAO[3]事件中黑客利用智能合约中的重入漏洞窃取360万以太币；2017年，Parity钱包提供的智能合约代码存在代码注入漏洞导致上亿美元资金被冻结[3]。智能合约已成为区块链安全的重灾区，每隔几个月就有新的智能合约漏洞被发现。

智能合约具有在区块链上交易公开透明、可追溯和不可篡改的优势，然而也因此引入了更多的攻击面。比如，导致The DAO事件的重入漏洞，就是利用了智能合约的公开性，使恶意攻击者能够递归调用受害合约的回退函数(fallback function)重复获得转账收益。此外，智能合约还有如短地址攻击、特权函数暴露、代码注入等许多漏洞。据研究统计，在以太坊中只有1%的智能合约是开源的[4]，绝大多数智能合约是以字节码的形式运行在以太坊虚拟机（EVM）上，且一旦上链就不可更改，因此研究基于字节码的漏洞检测技术具有重大意义。

传统智能合约漏洞检测方法主要受到代码漏洞检测方法的启发，依靠符号执行[5]和动态分析方法[6]进行漏洞检测。主流的智能合约的漏洞检测工具如Oyente[7]，是最早的智能合约漏洞分析工具之一，使用符号执行方法分析智能合约字节码漏洞，也常被作为其他检测方法（例如Maian[8]和Osiris[9]）的基础。Mythril[10]依靠对智能合约字节码的符号执行，控制流分析和污点检测进行智能合约漏洞挖掘。Slither[11]是一个静态分析框架，它将智能合约转换为名为SlithIR的中间表示，并应用数据流和污点跟踪等程序分析技术进行漏洞挖掘。Smartcheck[12]是基于智能合约源代码进行词法和语法分析的静态分析工具，用于查找漏洞模式和编程中的错误。Manticore[13]使用符号执行来查找智能合约字节码中可能导致重入漏洞和潜在自毁操作的执行路径。

这些方法存在以下不足：1）自动化程度较低，对智能合约的漏洞分析依赖于分析人员自身的技术水平与分析经验。2）检测准确率较低，误报率较高，覆盖漏洞种类不全面。Durieux等[14]使用经过手工分类的智能合约数据集对这些工具进行了测试，发现这些工具对漏洞的检测率最高仅能达到27%，且无法覆盖所有漏洞种类。在使用过程中，检测工具的误报率及各类警报较多。3）拓展性差，适用性低。传统方法检测工具严重依赖专家规则，但专家规则适用性有限，无法覆盖数量迅速增长的智能合约，且过多的规则会导致检测器庞大且效率低下[14]。

近年来，基于图神经网络的方法在传统代码漏洞检测方向取得了很大成效[15−16]。图神经网络由Scarselli等提出[17]，通过更改神经网络模型和消息传递方式，增加深度学习组件，能够针对不同数据的特点实现具有不同功能的神经网络模型。例如：图卷积神经网络（GCN）[18]利用卷积层来更新节点嵌入；GraphSAGE[19]能够利用节点属性信息产生未知节点的归纳式学习框架；图注意力网络GAT[20]能够通过自注意力机制对邻居节点进行聚合，实现了权重自适应分配。Xu等[21]提出了一种基于图神经网络的二进制代码相似度比较方法Gemini，展示出了图神经网络在漏洞挖掘领域的巨大潜力。Xu等[22]已经证明图神经网络具备足够通过Weisfeiler–Lehman测试的识别能力，即能在多项式时间内解决同构图问题，一定程度上解决了传统基于代码匹配和符号执行的方法执行效率低下的问题。

上述方法虽然解决了传统方法自动化程度低的问题，但仍具有以下不足：1）缺乏对智能合约字节码语义特征的提取，检测效果不佳。基于图神经网络的智能合约漏洞检测方法主要考虑图结构而缺少对智能合约字节码的语义特点进行特征提取，只考虑了神经网络对函数调用关系的学习，导致节点中大量

语义信息丢失[23]，难以生成高质量节点表征（高质量的节点表征能够用于衡量节点相似性，同时也是准确分类节点的前提）。2）部分检测方法覆盖的漏洞类型较少，对于新出现的漏洞不具有很好的适应性。

为了解决上述智能合约漏洞检测方法存在的问题，本文提出一种基于语义感知的图神经网络智能合约漏洞检测方法（L–GCN），针对实际部署环境下的智能合约，解决漏洞检测方法准确率低，误报率高，难以覆盖复杂的智能合约函数调用关系的问题。该方法提出基于智能合约字节码的语义提取方式，将自然语言处理的方法应用于智能合约字节码指令的语义提取，使用语义向量代替低维向量，生成了更高质量的节点表征；利用GCN处理非欧几里得结构样本时具有良好性能的优势，结合语义信息进行漏洞检测，增强了对节点特征的学习；通过实验设计说明了语义特征对于智能合约漏洞检测的重要性。

1　方法设计

本文提出的方法基于智能合约字节码实现。在以太坊中，绝大多数智能合约以字节码的形式运行，因此，从字节码提取语义信息，并结合图神经网络进行漏洞检测的方法，更适应于智能合约的运行环境。方法整体框架如图1所示。

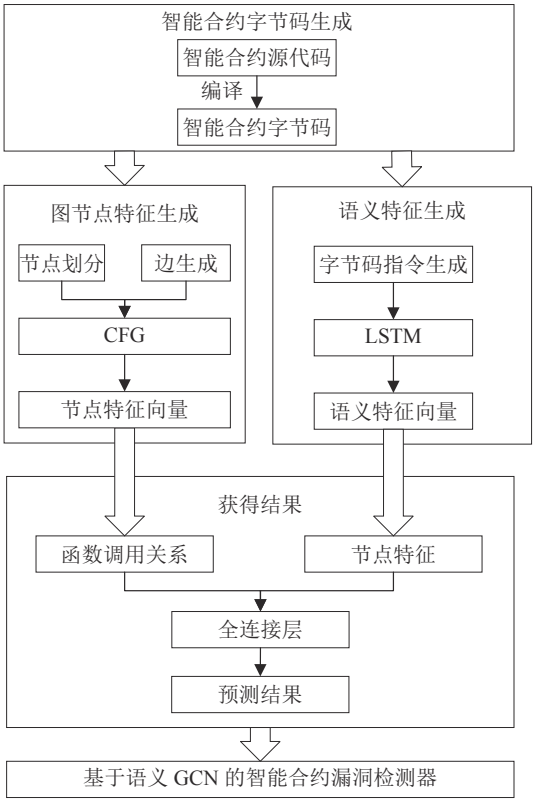

图 1　语义感知GCN智能合约漏洞识别框架

Fig. 1　Semantic-aware GCN smart contract vulnerability identification framework

本文方法总体架构包括4个阶段：1）智能合约字节码生成阶段，基于公开数据集中智能合约源代码，根据合约代码内声明的编译器版本进行编译，生成字节码；2）图节点特征生成阶段，从字节码中划分节点，生成CFG；并将其输入GCN中进行训练，生成节点特征。3）语义特征生成阶段，对每个节点内的字节码指令提取语义信息后生成的词向量，输入LSTM[24]网络中训练生成语义特征。4）获得结果阶段，将节点特征与语义特征的向量表示进行拼接后输入全连接层，映射到样本标记空间，并得到最终的预测结果。下面对4个阶段进行详细介绍。

1.1　智能合约字节码生成

智能合约在以太坊上部署时会以字节码的形式运行，因此生成智能合约字节码作为漏洞检测的基础更符合实际情况。但当前智能合约的公开数据集均以源代码形式发布，因此需要对数据集进行编译来生成字节码。

智能合约的编译器版本非常复杂，不同版本的编译器互相不兼容，且编译后的字节码也有一定的差异。本文在进行编译的过程中严格按照每个智能合约源代码中声明的编译器版本进行编译，防止出现因编译器版本不同产生的字节码不同导致对检测效果的干扰。

1.2　图节点特征提取

CFG包含智能合约结构信息，能够表示函数间的调用关系。本文基于智能合约字节码指令对节点进行重新划分，能够使函数调用关系更加清晰，使GCN更好地从CFG中学习控制特征。Allamanis等[25]研究表明，程序可以转换为符号图表示，符号图表示能够保留程序元素之间的语义关系。Huang等[26]研究表明，在智能合约漏洞识别过程中，不同函数的重要性不同。受此启发，本文从智能合约源代码中利用函数调用关系重新划分节点生成CFG。每个智能合约生成的CFG由合约中的节点以及节点之间的边表示，下面分别描述如何获得节点以及边。

在对数据集中的源代码进行编译生成字节码后，将字节码进行反汇编生成字节码指令。在确定函数入口后，只需要找到函数的结束指令就能按照函数划分出一个基本块。对智能合约字节码指令进行分析得到表示基本块结束的指令总结如表1所示。每个节点表示一个基本块，每个基本块不一定覆盖一个完整函数，因为函数内部也存在指令跳转。使用集合V表示所有CFG中的节点。

在获得节点之后，将节点之间的调用关系看作一条边，代表前一个函数可能会调用下一个函数。每个节点可能调用多个节点，也可能被多个节点调用。

表 1 代表基本块结束的指令

Tab. 1 End instructions of basic block

指令	作用
STOP	停止
SELFDESTRUCT	自毁
RETURN	返回
REVERT	判断
INVALID	无效
SUICIDE	删除
JUMP	跳转
JUMPI	跳转

使用集合E表示CFG中的边。

经过节点划分和边构造后，使用G表示CFG，则$G=(V,E)$。将CFG输入GCN中进行训练，获得图的特征。对于每一层神经网络，都可以用如下的非线性函数表示：

$$\boldsymbol{H}^{(l+1)}=f(\boldsymbol{H}^{(l)},\boldsymbol{A}) \tag{1}$$

式中：$l\in[0,L]$，其中L为网络层数；输入层$\boldsymbol{H}^{(0)}=\boldsymbol{X}$，其中$\boldsymbol{X}$为特征矩阵；输出层$\boldsymbol{H}^{(L)}=\boldsymbol{Z}$，其中$\boldsymbol{Z}$为节点水平的输出；$f(\cdot)$为神经网络的可微函数；$\boldsymbol{A}$为$G$的邻接矩阵。GCN对图的分类利用了节点自身的特征信息及图的结构信息，学习策略如下：

$$\Gamma=\Gamma_0+\lambda\Gamma_{\text{reg}} \tag{2}$$

$$\Gamma_{\text{reg}}=f(\boldsymbol{X})^{\mathrm{T}}\Delta f(\boldsymbol{X}) \tag{3}$$

式中：Γ_0为图中带标签节点的监督损失，Γ_{reg}为图结构信息引入的损失，λ为权重系数，$\boldsymbol{X}$为节点特征向量矩阵，Δ表示图的拉普拉斯算子。

GCN模型的分层传播规则如下：

$$f(\boldsymbol{H}^{(l+1)},\boldsymbol{A})=\sigma(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}) \tag{4}$$

式中：$\widetilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}$，$\boldsymbol{I}$为单位矩阵，$\widetilde{\boldsymbol{A}}$为加入自循环的邻接矩阵，用于将节点自身信息与所有邻居节点进行聚合；$\widetilde{\boldsymbol{D}}$为$\widetilde{\boldsymbol{A}}$矩阵的度矩阵；$\boldsymbol{W}^{(l)}$为第$l$层神经网络层的权重矩阵；$\sigma(\cdot)$表示线性整流函数（linear rectification function, ReLU）等非线性激活函数。

1.3 节点语义信息提取

第1.2节生成的CFG图提供了图结构信息，CFG图的节点是智能合约的基本块，每个基本块由一系列指令构成，需要从这些指令中提取特征。基于人工选择特征的低纬度嵌入会导致大量语义信息的丢失[23]。使用自然语言处理（natural language processing, NLP）模型提取这些指令作为节点特征能够最大程度地保留语义信息。

在通过节点划分及字节码反汇编获得每个节点的字节码指令后，将字节码指令视为自然语言处理。首先，根据指令进行分词；然后，将序列映射为向量表示，即将词语在词语集合中的下标作为词的表示；最后，将指令向量序列输入LSTM网络中进行训练得到语义表征。

考虑到指令信息之间的时序性、连贯性，使用LSTM网络能够解决梯度消失问题，学习到指令间的长距离依赖关系，使指令之间的顺序性在序列中充分体现，最大程度地保留语义特征。

LSTM使用门控机制，由输入门、遗忘门和输出门作为一个模块，由多个结构相同的模块串联形成，当指令序列顺序通过LSTM网络时，这些模块中的门结构会进行调节需要记忆和遗忘的特征，从而得到最终生成的具有长距离依赖的语义特征。门控机制的具体公式如下[24]：

$$\boldsymbol{f}_t=\sigma(\boldsymbol{W}_f\boldsymbol{x}_t+\boldsymbol{U}_f\boldsymbol{h}_{t-1}+\boldsymbol{b}_f) \tag{5}$$

$$\boldsymbol{i}_t=\sigma(\boldsymbol{W}_i\boldsymbol{x}_t+\boldsymbol{U}_i\boldsymbol{h}_{t-1}+\boldsymbol{b}_i) \tag{6}$$

$$\boldsymbol{o}_t=\sigma(\boldsymbol{W}_o\boldsymbol{x}_t+\boldsymbol{U}_o\boldsymbol{h}_{t-1}+\boldsymbol{b}_o) \tag{7}$$

$$\boldsymbol{c}=\tan h(\boldsymbol{W}_c\boldsymbol{x}_t+\boldsymbol{U}_c\boldsymbol{h}_{t-1}+\boldsymbol{b}_c) \tag{8}$$

$$\boldsymbol{c}_t=\boldsymbol{f}_t\circ\boldsymbol{c}_{t-1}+\boldsymbol{i}_t\circ\boldsymbol{c} \tag{9}$$

$$\boldsymbol{h}_t=\boldsymbol{o}_t\circ\tan\boldsymbol{h}(\boldsymbol{c}_t) \tag{10}$$

式（5）～（10）中：$\boldsymbol{x}$和$\boldsymbol{h}$分别为输入和输出向量；$\sigma(\cdot)$为sigmoid函数；t为时间步长值；$\boldsymbol{W}$为输入的权重；$\boldsymbol{U}$为循环输出的权重；$\boldsymbol{b}$为偏差向量；运算符“$\circ$”为逐元素乘法；$\boldsymbol{c}$为存储单元；$\boldsymbol{f}_t$为遗忘门向量，控制上一个时刻内部状态$\boldsymbol{c}_{t-1}$需要遗忘的信息；输入门向量$\boldsymbol{i}_t$控制当前的内部状态$\boldsymbol{c}_t$要保留多少信息；输出门向量$\boldsymbol{o}_t$控制当前内部状态$\boldsymbol{c}_t$需要输出哪些信息给外部状态$\boldsymbol{h}_t$。

1.4 结合特征获得预测结果

获得节点特征和语义信息特征后，将两种特征进行拼接，输入到全连接层中将两种特征综合起来，映射到样本标记空间。

为了减小图结构数据在分类模型中产生的同构偏差[27]，获得全连接层的输出后，将其输入逻辑回归层（softmax layer）进行归一化：

$$\text{softmax}(g_i)=\frac{\exp(g_i)}{\sum_{z=1}^{Z}\exp(g_n)} \tag{11}$$

式中：g_i为第i个节点的输出值；z为输出节点的个数；通过softmax函数可以将输出转化为概率分布，完成预测的目标。

检测无漏洞为正类样本，有漏洞为负类样本，通过与标签对比会有4种情况，即：TP表示将正类预测为正类，FN表示将正类预测为负类，FP表示将负类

预测为正类，TN表示将负类预测为负类。采用n表示样本数量，准确率（accuracy）表示样本中正类与负类分类正确的比例，精确率（precision）表示在预测为正类的样本中真正的正类样本所占的比例，召回率（recall）表示在所有真正的正类样本中被正确预测的比例，F1分数（F1 score）用于衡量精确率和召回率之间的平衡性。4种指标表达式分别为：

$$F_{\mathrm{acc}} = (n_{\mathrm{TP}} + n_{\mathrm{TN}})/(n_{\mathrm{TP}} + n_{\mathrm{FN}} + n_{\mathrm{FP}} + n_{\mathrm{FN}}) \quad (12)$$

$$F_{\mathrm{pre}} = n_{\mathrm{TP}}/(n_{\mathrm{TP}} + n_{\mathrm{FP}}) \quad (13)$$

$$F_{\mathrm{rec}} = n_{\mathrm{TP}}/(n_{\mathrm{TP}} + n_{\mathrm{FN}}) \quad (14)$$

$$F_1 = 2 \cdot F_{\mathrm{pre}} \cdot F_{\mathrm{rec}}/(F_{\mathrm{pre}} + F_{\mathrm{rec}}) \quad (15)$$

式（12）～（15）中，F_{acc}为准确率，F_{pre}为精确率，F_{rec}为召回率，F_1为F1分数。

2　实验评估

2.1　数据集

使用Durieux等[14]收集的47 518个真实智能合约数据集SmartWild及手工构造的具有漏洞的智能合约数据集SBcurated。

SmartWild[14]数据集收集自真实世界发生过交易的智能合约，这些合约经过重复性筛选，是具有潜在漏洞的智能合约，使用真实的以太坊合约地址作为唯一标志。由于智能合约编译器版本非常多，且各个版本之间并不兼容，导致从源代码编译生成的字节码有很大的差异，对实验结果会产生影响。许多编译器版本较低的智能合约已经很少运行在以太坊上。为了更好地模拟现在的智能合约情况，本文在使用数据集时，过滤掉了一些编译器版本过早的智能合约，并对其余合约进行了字节码的重复性筛选，最终使用其中的21 437个智能合约进行实验。

SBcurated[14]数据集由一部分真实世界中具有漏洞的合同和一部分手工构造的具有漏洞的合同组成。该数据集中的智能合约被手工标记了漏洞的位置和类别，可用于评估智能合约分析工具识别漏洞的有效性。该数据集共有136个样本。

本文在实验部分使用上述两种数据集进行漏洞检测，并设计对比实验，展示语义信息对于智能合约漏洞检测的有效提升以及检测真实世界中具有漏洞的智能合约的能力。

2.2　实验设计

首先，文献[14]的研究表明，Mythril[10]与Slither[11]的组合使用兼具准确性和效率，因此选择这两种方法的组合对SmartWild数据集21 437个智能合约生成标签。使用该数据集进行训练和测试，其中，训练集数量为19 437，测试集数量为2 000。使用本文方法（L–GCN）对这些智能合约进行检测，与Oyente[7]、Smartcheck[12]、Manticore[13]3种传统检测工具以及一种仅使用GCN网络[18]不含语义特征的漏洞检测方法（简称GCN）进行对比。使用的评价指标有准确率（accuracy）、精确率（precision）、召回率（recall）和F1分数（F1 score）。

然后，使用Sbcurated数据集中136个手工分类的智能合约进行测试，用于测试本文方法（L–GCN）与传统的Oyente[7]、Smartcheck[12]、Manticore[13]及GCN[18]检测方法发现真实智能合约漏洞的能力。由于该数据集中所有合约均存在漏洞，因此仅对比准确率。

2.3　实验结果分析

2.3.1　SmartWild数据集实验结果分析

表2展示了使用5种方法对SmartWild数据集21 437个真实智能合约进行智能合约漏洞检测的性能指标结果。

表 2　基于SmartWild数据集的性能比较

Tab. 2　Performance comparison based on SmartWild dataset

检测方法	准确率/%	精确率/%	召回率/%	F1分数/%
Manticore	36.57	31.90	86.32	46.58
Oyente	37.36	33.13	92.73	48.82
Smartcheck	39.72	32.53	81.01	31.09
GCN	68.05	67.42	88.73	76.62
L–GCN	81.40	79.23	92.79	85.48

根据表2中的实验数据，从5种方法的对比中可以看出，本文提出的L–GCN在4种指标上均超过其余4种方法。在4种指标中，L–GCN方法在准确率、精确率以及F1分数上都有较大的提升，但5种方法的召回率差距不大，表明5种方法都更注重对于具有漏洞的智能合约的检出率。前4种方法的精确率较低，说明这些方法在检测过程中有较高的误报率；而L–GCN方法在拥有最高的召回率值同时，提升了精确率值，说明该方法能够有效降低误报率。与GCN方法相比，L–GCN方法的4个指标均有所上升，说明了增加语义信息确实能提升漏洞检测效果。

值得一提的是，Oyente方法拥有极高的召回率值，几乎与L–GCN方法相同，推测可能的原因是，SmartWild数据集使用基于符号执行的智能合约漏洞检测方法的组合打标签，对于同样基于符号执行的Oyente方法有一定提升。

该实验结果表明，加入语义特征能够有效提升智能合约漏洞检测效果，而本文提出基于语义感知图神经网络的智能合约漏洞检测方法能够实现提升

检测精度且降低误报率的目标。

2.3.2 SBcurated数据集实验结果分析

表3展示了使用5种方法对SBcurated数据集中136个智能合约进行漏洞检测的准确率及检测出含有漏洞的智能合约数。

表 3 SBcurated数据集准确率

Tab. 3 Accuracy comparison based on SBcurated dataset

检测方法	准确率/%	含有漏洞的智能合约数
Manticore	79.17	108
Oyente	80.88	110
Smartcheck	71.37	97
GCN	87.50	119
L–GCN	92.65	126

与SmartWild数据集相比，SBcurated数据集虽然数据量较小，但其中的智能合约均采用手工分类，标签的准确率更高，能够更好地展现检测方法的效果。5种检测方法中，Manticore由于检测时间超时仅检测成功48个样本并生成了结果，其余4种方法均完成全部样本的检测。

从表3的指标来看，本文提出的基于语义感知图神经网络的智能合约漏洞检测方法（L–GCN）的检测准确率和含漏洞的数目均高于其他4种方法，但提升的效果不如SmartWild数据集的明显。推测可能的原因是，SmartWild数据集中样本均有漏洞，因此均为正类样本，此时，准确率实际上等同于表2中的召回率指标。对比表2中召回率指标和表3的准确率后也发现，这5种检测方法在这两个指标上的排名几乎相同。这表明，5种方法在对有漏洞的合约上都有着较高的检出率，当样本中据有漏洞的合约较多时检出效果更好，而本文提出的L–GCN方法比其余4种方法依旧有较大提升。

通过对检出有漏洞的合约进行统计，L–GCN方法检测出的所有具有漏洞的合约中，有4个具有漏洞的合约是前4种方法均未能检测出的。

该实验表明，在检测真实智能合约漏洞时，本文提出的基于语义感知图神经网络的智能合约漏洞检测方式的检测效果更好，并且能够检测出实验中其他方法未能检出的含有漏洞的合约。

3 总结与展望

本文提出了一个基于语义感知图神经网络的智能合约漏洞分析方法，与现有方法相比，将神经网络对图结构良好的处理能力与自然语言处理方法对语义信息的提取相结合，在检测漏洞的过程中既使用函数调用关系也学习节点自身字节码特征，探索了语义信息加入神经网络进行智能合约漏洞检测的可能性。经过大量实验表明，基于语义感知图神经网络的智能合约漏洞检测方法能够有效提升检测精度，降低误报率，并且具有检测出真实智能合约中漏洞的能力。本文的工作展示了语义信息和神经网络结合的方法在智能合约漏洞检测方向的潜力，具有继续研究的意义。在未来的研究工作中，将研究各类智能合约漏洞具体特点，对每类漏洞更具针对性地进行语义信息提取，以达到更细致分类的目的。

参考文献：

[1] Ni Yuandong,Zhang Chao,Yin Tingting.A survey of smart contract vulnerability research[J].Journal of Cyber Security,2020,5(3):78–99.[倪远东,张超,殷婷婷.智能合约安全漏洞研究综述[J].信息安全学报,2020,5(3):78–99.]

[2] Sankar L S,Sindhu M,Sethumadhavan M.Survey of consensus protocols on blockchain applications[C]//Proceedings of the 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS).Coimbatore:IEEE,2017:1–5.

[3] Fu Menglin,Wu Lifa,Hong Zheng,et al.Research on vulnerability mining technique for smart contracts[J].Journal of Computer Applications,2019,39(7):1959–1966.[付梦琳,吴礼发,洪征,等.智能合约安全漏洞挖掘技术研究[J].计算机应用,2019,39(7):1959–1966.]

[4] Fröwis M,Böhme R.In code we trust?[M]//Data Privacy Management,Cryptocurrencies and Blockchain Technology.Cham:Springer,2017:357–372.

[5] Shigemura R A L,Gonçalves G S,Oliveira F A,et al.Wibx:Making smart contracts even smarter[C]//WAIAF 2019 –Workshop of Artificial Intelligence Applied to Finance.São José dos Campos:WAIAF,2019:1–7.

[6] Jiang Bo,Liu Ye,Chan W K.ContractFuzzer:Fuzzing smart contracts for vulnerability detection[C]//Proceedings of the Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering.New York:ACM,2018:259–269.

[7] Luu L,Chu D H,Olickel H,et al.Making smart contracts smarter[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.New York:ACM,2016:254–269.

[8] Nikolić I,Kolluri A,Sergey I,et al.Finding the greedy,prodigal,and suicidal contracts at scale[C]//Proceedings of the 34th Annual Computer Security Applications Conference.New York:ACM,2018:653–663.

[9] Torres C F,Schütte J,State R.Osiris:hunting for integer bugs in ethereum smart contracts[C]//Proceedings of the 34th Annual Computer Security Applications Conference.New York:ACM,2018:664–676.

[10] Mueller B.Mythril—Reversing and bug hunting framework for the ethereum blockchain[EB/OL].[2021–08–31]. https://pypi.org/project/mythril/0.8.2/.

[11] Feist J,Grieco G,Groce A.Slither:A static analysis framework for smart contracts[C]//Proceedings of the 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB).Montreal:IEEE,2019:8–15.

[12] Tikhomirov S,Voskresenskaya E,Ivanitskiy I,et al.SmartCheck:static analysis of ethereum smart contracts[C]//Proceedings of the 2018 IEEE/ACM 1st International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB).Gothenburg:IEEE,2018:9–16.

[13] Mossberg M,Manzano F,Hennenfent E,et al.Manticore:A user-friendly symbolic execution framework for binaries and smart contracts[C]//Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE).San Diego:IEEE,2019:1186–1189.

[14] Durieux T,Ferreira J F,Abreu R,et al.Empirical review of automated analysis tools on 47,587 ethereum smart contracts[C]//Proceedings of the 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). Seoul:IEEE,2020:530–541.

[15] Cheng Zhiyong,Chang Xiaojun,Zhu Lei,et al.MMALFM: Explainable recommendation by leveraging reviews and images[EB/OL].[2021–08–31].https://arxiv.org/abs/1811.05318.

[16] Liu Anan,Xu Ning,Zhang Hanwang,et al.Multi-level policy and reward reinforcement learning for image captioning[C]//Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence.Stockholm:International Joint Conferences on Artificial Intelligence Organization,2018:821–827.

[17] Scarselli F,Gori M,Tsoi A C,et al.The graph neural network model[J].IEEE Transactions on Neural Networks, 2009,20(1):61–80.

[18] Kipf T N,Welling M.Semi-supervised classification with graph convolutional networks[EB/OL].[2021–08–31]. https://arxiv.org/abs/1609.02907.

[19] Xu Da,Ruan Chuanwei,Korpeoglu E,et al.Inductive representation learning on temporal graphs[EB/OL].[2021–08–31].https://arxiv.org/abs/2002.07962.

[20] Vaswani A,Shazeer N,Parmar N,et al.Attention is all you need[C]//Advances in Neural Information Processing Systems 30 (NIPS 2017).Long Beach:NIPS,2017:5998–6008.

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

[22] Xu Keyulu,Hu Weihua,Leskovec J,et al.How powerful are graph neural networks?[EB/OL].[2021–08–31].https://arxiv.org/abs/1810.00826.

[23] Yu Zeping,Cao Rui,Tang Qiyi,et al.Order matters:Semantic-aware neural networks for binary code similarity detection[J].Proceedings of the AAAI Conference on Artificial Intelligence,2020,34(1):1145–1152.

[24] Hochreiter S,Schmidhuber J.Long short-term memory[J].Neural Computation,1997,9(8):1735–1780.

[25] Allamanis M,Brockschmidt M,Khademi M.Learning to represent programs with graphs[EB/OL].[2021–08–31].https://arxiv.org/abs/1711.00740.

[26] Huang Jianjun,Han Songming,You Wei,et al.Hunting vulnerable smart contracts via graph embedding based bytecode matching[J].IEEE Transactions on Information Forensics and Security,2021,16:2144–2156.

[27] Ivanov S,Sviridov S,Burnaev E.Understanding isomorphism bias in graph data sets[EB/OL].[2021–08–31].https://arxiv.org/abs/1910.12091.

（编辑　赵　婧）

引用格式：Zhao Bo,Shangguan Chenhan,Peng Xiaoyan,et al.Semantic-aware graph neural network for smart contract bytecode vulnerability detection[J].Advanced Engineering Sciences,2022,54(2):49–55.[赵波,上官晨晗,彭小燕,等.基于语义感知图神经网络的智能合约字节码漏洞检测方法[J].工程科学与技术,2022,54(2):49–55.]