基于预训练与新型时序图神经网络的智能合约漏洞检测方法

庄园[1]，樊泽楷[1]，王诚[1]，孙建国[2]，李耀麟[3]
（1.哈尔滨工程大学计算机科学与技术学院，黑龙江 哈尔滨 150001；2.西安电子科技大学杭州研究院，浙江 杭州 311231；
3.北京理工大学计算机学院，北京 100081）

摘　要：针对现有深度学习漏洞检测方法对合约字节码特征挖掘不足、漏洞语义表征不精准，且传统图神经网络模型对合约语句的时序信息学习能力不足，提出一种基于预训练与时序图神经网络的智能合约漏洞检测方法。首先，通过预训练模型将智能合约字节码建模为漏洞语义感知的合约图结构。其次，结合自注意力机制，设计了一种新颖的基于事件驱动的时序图神经网络模型，实现对合约执行中时序信息的有效抽取。最后，聚焦于可重入漏洞、时间戳依赖漏洞以及 Tx.origin 身份认证漏洞，通过 120 932 份真实合约数据集进行大量的评估实验，结果表明所提方法的检测效果显著优于现有方法。

关键词：区块链；智能合约；漏洞检测；预训练模型；图神经网络
中图分类号：TP309.2
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2024163

Smart contract vulnerability detection method based on pre-training and novel timing graph neural network

ZHUANG Yuan[1], FAN Zekai[1], WANG Cheng[1], SUN Jianguo[2], LI Yaolin[3]
1. Collage of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
2. Hangzhou Institute of Technology, Xidian University, Hangzhou 311231, China
3. School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Abstract: To address the limitations of current deep learning-based methods in extracting contract bytecode features and representing vulnerability semantics, as well as the shortcomings of the traditional graph neural networks in learning temporal information from contract statements, a method for detecting vulnerabilities in contracts was proposed based on pre-trained and temporal graph neural network. Firstly, the pre-trained model was used to transform smart contract bytecode into a vulnerability semantics-aware contract graph structure. Then, combined with a self-attention mechanism, the event-driven temporal graph neural network was designed to extract temporal information during contract execution. Finally, focusing on reentrant vulnerabilities, timestamp dependency vulnerabilities, and Tx.origin authentication vulnerabilities, extensive experiments were conducted on a dataset of 120 932 actual contracts. The results show that the proposed method significantly outperforms existing approaches.

Keywords: blockchain, smart contract, vulnerability detection, pre-training model, graph neural network

0　引言

近年来，随着区块链技术的不断发展，以太坊中智能合约数量不断增多，智能合约已经在金融、医学、供应链、版权保护等多个领域得到了广泛应

收稿日期：2024–02–18；**修回日期：**2024–08–06
基金项目：国家自然科学基金资助项目（No.62202121）；国家重点研发计划基金资助项目（No.2022YFB4400703）；中央高校基本科研业务费专项资金资助项目（No.3072022TS0604）
Foundation Items: The National Natural Science Foundation of China (No.62202121), The National Key Research and Development Program of China (No.2022YFB4400703), The Fundamental Research Funds for the Central Universities (No.3072022TS0604)

用。用户在享受智能合约带来便利服务的同时，也面临着它带来的安全威胁。由于智能合约运行在不具有可信计算环境的底层基础设施上，并且具有区别于传统程序的特性，例如智能合约被部署上线后便无法进行修改，若遭到恶意攻击，其漏洞不能被删除或更改修复，导致用户资产的经济损失无法挽回。至今已发生多起针对智能合约的安全攻击事件，不仅给用户造成了巨大的经济损失，同时也导致用户对区块链技术的安全性产生怀疑。因此，智能合约的漏洞检测工作正受到越来越多的关注[1-2]。

传统的智能合约漏洞检测技术包括形式化验证、符号执行、静态分析、模糊测试等。形式化验证是一种有效的验证程序是否符合预期设计属性和安全规范的技术。例如，以太坊基金会Hirai等[3]利用Isabelle/HOL对以太坊虚拟机中的指令语义进行形式化描述，以将其用于手动证明某个程序的安全性。符号执行是一种传统的自动化漏洞挖掘技术，符号执行策略将程序所需的外部输入抽象为取值不固定的符号值，并通过不断求解路径约束，来尽可能地探索程序分支。作为最早的智能合约漏洞检测工具之一，Oyente[4]利用符号执行技术，通过智能合约字节码和以太坊状态模拟不同的执行路径。Slither[5]提出使用静态分析进行漏洞检测，聚焦于指令、语法等结构信息的还原推理上。但是这些传统的检测方法严重依赖一些固定规则，并且无法分析识别逻辑复杂的合约漏洞，导致检测准确率较低，存在较高的误报率和漏报率。不同于静态方法，Contractfuzzer[6]基于模糊测试技术，以动态执行方法实现智能合约漏洞检测，但这类动态方法受限于用户设计和提供测试用例的效果，而且很难通过算法探索所有可能的执行路径。

随着人工智能在程序分析领域的兴起，近年来研究者们[7]开始尝试使用深度学习技术解决智能合约安全漏洞问题。现有漏洞检测方法主要面向智能合约源码，其代码形式简单直接，语义信息相对完整。然而，部署在区块链上的智能合约均以字节码形式表示，超过98%的智能合约仅提供合约字节码形式。因此，在实际应用中面向智能合约字节码的漏洞检测方案更为重要。目前，只有少数工作针对智能合约字节码进行合约安全检测。现有的基于神经网络的方法无法深入挖掘漏洞与底层以太坊虚拟机（EVM, ethereum virtual machine）指令间的关系，尤其是不能充分捕获代码执行的时序信息，导致检测准确率较低。此外，由于智能合约字节码存在同质性，缺少智能合约漏洞关键指令的识别将导致检测方法容易受到漏洞逻辑无关代码的干预，极易影响漏洞的检测精度。

为解决上述问题，本文提出一种基于预训练与时序图神经网络的智能合约漏洞检测方法。具体而言，本文将智能合约字节码表征为全局语义合约控制流程图，根据漏洞语义的关键指令抽取核心图结构，将合约中无关的代码信息对模型的影响降到最低。本文将智能合约核心图传入预训练模型来自Transformer的双向编码器表示（BERT, bidirectional encoder representation from transformer）进行节点编码，并利用基于事件驱动的时序图神经网络（ETGN, event-based temporal graph neural network）将合约图中节点信息、边信息以及时序信息进行事件构造，结合自注意力机制以提高程序执行中时序事件的并发性，最终将合约图语义表征应用于智能合约漏洞检测，从而显著提升智能合约漏洞识别的准确度。

本文的主要贡献如下。

1) 提出了一个面向智能合约字节码的漏洞检测框架，该框架包含数据预处理、核心图构建、预训练模型以及基于事件驱动的时序图神经网络4部分，实现了端到端的智能合约漏洞检测，有效提高了漏洞检测效率，减少了漏报和误报情况。

2) 提出了一种新颖的基于事件驱动的时序图神经网络并应用于智能合约漏洞检测。该模型基于事件进行驱动，并结合注意力机制提高时序事件的并发性，有效提高了合约深层语义信息的挖掘。

3) 本文在包含超过10万份真实智能合约的数据集上进行实验，对于可重入漏洞、时间戳依赖漏洞以及Tx.origin身份认证漏洞检测的准确率分别达到94.67%、93.08%、95%。大量对比实验的结果表明，本文方法的性能优于传统方法以及最新提出的基于神经网络的漏洞检测方法。

1 相关工作

本节将对智能合约漏洞检测的相关工作进行介绍，主要包括传统的智能合约漏洞检测方法和基于神经网络的智能合约漏洞检测方法两类。

1.1 传统的智能合约漏洞检测方法

早期研究者们普遍采用符号执行、静态分析、

形式化验证等传统的智能合约漏洞检测方法。例如，基于符号执行的Oyente[4]以智能合约字节码和以太坊状态作为输入，模拟EVM从而遍历合约的不同执行路径，其支持检测的漏洞类型包括可重入漏洞、异常处理漏洞、交易顺序依赖漏洞等；另一采用符号执行的合约分析工具Mythril[8]则将字节码解析为中间形式表示，使用符号执行来模拟合约的执行路径，但对于复杂的智能合约，其需要较长时间完成分析，且具有较高的漏报率。基于静态分析的Securify[9]特别关注合约的语义属性，通过属性来判断合约是否满足或违反特定的安全规范，但此工具受限于预定规则，若规则制定不完整或过时，将对漏洞检测精度造成影响。基于形式化验证的ZEUS[10]则利用约束语句和符号模型快速验证合约的安全性，但存在自动化程度不高、通用性较差等问题；最新的工作Pluto[11]实现了跨合约的漏洞检测，其针对字节码建立跨合约控制流程图（ICFG, inter-contract control flow graph），通过符号执行对ICFG进行路径探索，但仍面临路径爆炸的挑战。综上，现有方法针对字节码合约的逆向分析技术仍聚焦于指令、语法等结构信息的还原推理，不足以表达合约语义，因此难以应用于分析漏洞及其作用机理。

1.2　基于神经网络的智能合约漏洞检测方法

近年来，深度学习与程序分析领域的融合诞生了各种新型的基于神经网络的智能合约漏洞检测方法[12-13]。Mi等[14]提出在控制流程图（CFG, control flow graph）上进行深度优先遍历得到具有智能合约执行顺序信息的操作码序列，并应用N-gram和词频-逆向文件频率（TFIDF, term frequency-inverse document frequency）将操作码序列转换为向量表示，再输入到神经网络中进行训练和分类。基于信息图和集成学习的智能合约漏洞检测方法[15]（SCVDIE, smart contract vulnerability detection method based on information graph and ensemble learning）从操作码中计算共现频率，构建操作码共现矩阵，基于共现关系构建信息图从而表示漏洞模式信息。Huang等[16]提出了一种基于图嵌入的字节码匹配方法，模拟执行操作码构造控制流图，跟踪与漏洞相关的数据流，并利用图嵌入方法计算切片间的相似性。Fan等[17]提出了一种双注意力图卷积神经网络（DA-GCN, dual attention graph convolution network）漏洞检测方法，利用CFG提取特征，并通过图卷积模块、双注意力模块和分类模块处理，有效检测可重入漏洞和时间戳依赖漏洞。上述针对智能合约字节码的漏洞检测方案，仅简单考虑了操作码语义信息，未深度分析CFG中与漏洞语句相关的节点和边，导致神经网络无法全面学习与推理漏洞语句的语义信息。

综上所述，传统的智能合约漏洞检测方法和现有的基于神经网络的智能合约漏洞检测方法尚未充分挖掘CFG中的时序信息以及漏洞语义信息。因此，本文提出一种基于预训练与时序图神经网络的智能合约漏洞检测方法，将合约操作码信息通过预训练模型BERT进行表征，并结合基于事件驱动的时序图神经网络，将漏洞语句相关信息进行事件建模，从而实现高精度的漏洞检测。

2　基于预训练与时序图神经网络的智能合约漏洞检测方法

本文提出的基于预训练与时序图神经网络的智能合约漏洞检测方法的方案框架如图1所示。该方案由4个部分组成：数据预处理，将字节码转换为CFG表示，并进行指令规范化；核心图生成，进行关键指令定义和核心图构建；预训练模型，主要完成指令向量化，利用BERT模型将节点进行语义表征；ETGN，基于事件驱动的时序图神经网络，实现合约图编码以完成高效的漏洞检测。

2.1　数据预处理

本文数据集来自以太坊平台公开的真实合约，针对智能合约的二进制代码，本文通过工具octopus[18]将其转化为CFG进行表示。CFG由普通节点和逻辑边组成，其中普通节点代表EVM指令集合，逻辑边代表节点之间的跳转关系。设$G=<V,E>$表示智能合约的控制流程图，V代表普通节点集合，E代表逻辑边集合，CFG转化示例如图2所示。

智能合约EVM指令中包含大量无意义参数，为了更好地描述数据特征，本文对普通节点的指令集进行规范，将指令中的具体数值替换为代表其数据属性的标签表达。比如指令的目标地址和内存地址的偏移量等，其参数与漏洞语义并无直接联系，

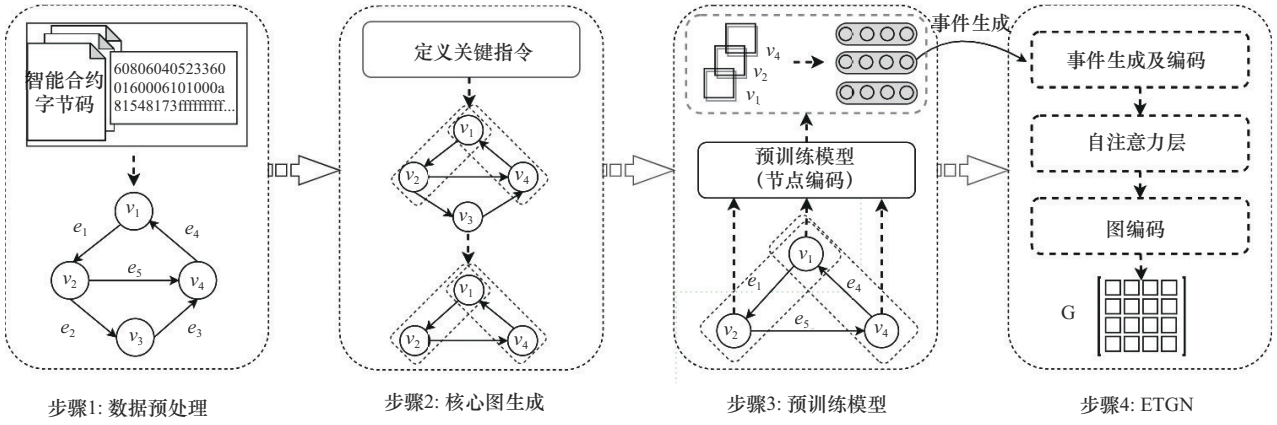


图1　基于预训练与时序图神经网络相关的智能合约漏洞检测方法的方案框架

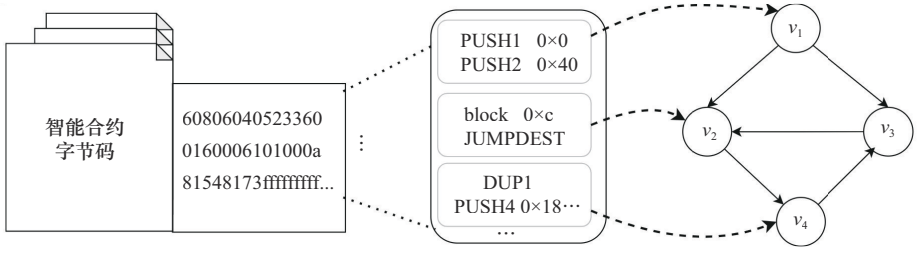


图2　CFG转化示例

若使用不同版本的编译器很可能会改变这些数据，这将为后续研究引入不必要的差异。

DUP类和SWAP类指令会对堆栈中的数据进行操作，但它们与源码行为关联不强，指令参数在漏洞检测中的作用可以忽略不计。此外，通过指令规范化可以减少预训练模型词典的规模，从而提升预训练模型的精度，表1列举了指令规范化示例。

表1　指令规范化示例

类别	指令	规范化后指令
DUP	DUP1, DUP2, ⋯, DUP16	DUP
SWAP	SWAP1, SWAP2, ⋯, SWAP16	SWAP
PUSH	PUSH1 xx, PUSH2 xx, ⋯, PUSH32 xx	PUSHx Datax

2.2　核心图生成

2.2.1　关键指令定义

传统CFG中包含了大量无关节点信息，容易对模型的漏洞语义学习产生较大干扰。通过对现有智能合约漏洞特点进行分析，本文根据不同漏洞的特征定义了关键指令，如表2所示。本文定义关键指令所在的节点为核心节点，依据核心节点有效构建核心图，从而将合约漏洞语义进行高效表征。

表2　关键指令

漏洞类型	关键指令	指令含义说明
可重入漏洞	CALL	调用外部账户
	MLOAD	从内存中加载数据
	SSTORE	将数据存储在内存
时间戳依赖漏洞	TIMESTAMP	获取块的时间戳
	LT	小于比较
	GT	大于比较
	CALL	调用外部账户
Tx.origin身份认证漏洞	ORIGIN	获取执行源地址
	MLOAD	从内存中加载数据

为了聚焦漏洞的语义特征，同时将无关指令信息进行过滤，本文最终给出符合漏洞语义的关键指令，这里关键指令可涵盖不同的智能合约漏洞种类。

以可重入漏洞为例，图3中合约DAO为一个具有可重入漏洞的智能合约案例，第8行call.value函数用于发送代币，涉及外部调用和加载数据等操作，第9行执行账户余额扣除，涉及状态变量更新。结合可重入漏洞的语法特征和汇编指令的信息，本文定义该漏洞的关键指令为CALL、MLOAD、SSTORE，它们分别标识了可重入漏洞的外部调用、数据加载和状态变量更新。基于上述推理过程，表2分别给出时间戳依赖漏洞以及Tx.origin身份认证漏洞的关键指令。

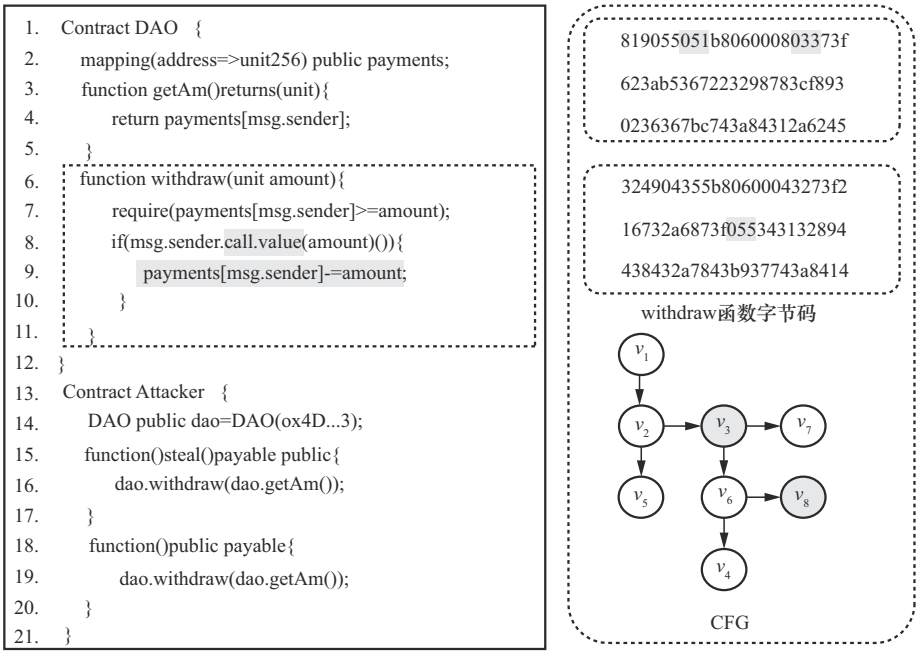


图3　可重入漏洞案例

2.2.2　核心图构建

根据节点与漏洞语义信息的相关程度，本文将CFG中的节点定义为4种类型，分别是调度节点、函数调用节点、核心节点和普通节点，具体定义如下。节点调用示例如图4所示。

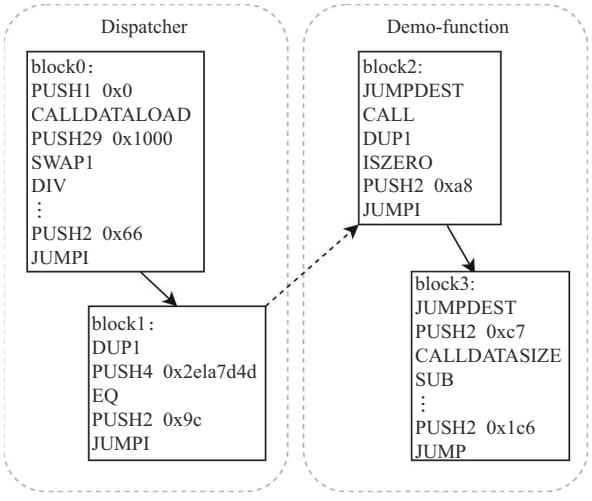


图4　节点调用示例

1) 调度节点：将合约字节码转化为CFG表示时，合约起始函数是Dispatcher函数，因此，本文将该节点定义为调度节点，也是合约CFG的首节点（即图4中block0）。

2) 函数调用节点：合约中的函数调用关系将在CFG中进行继承，如图4中block1即表示函数调用，所有满足PUSH、EQ、PUSH1、PUSH2、JUMPI这类指令序列的节点即为函数调用节点。

3) 核心节点：基于漏洞语义信息以及各个类型漏洞的特点，本文定义包含关键指令的节点为核心节点，如图4中block2（存在可重入漏洞关键指令CALL）。

4) 普通节点：除了以上几种节点类型外，智能合约CFG中其他block定义为普通节点。此处每个普通节点可涉及多个函数调用，如图4中block3所示。

根据节点间的跳转关系，本文将合约CFG的边分为3种类型：DIRCALL、CONDCALL、FUNCCALL，分别代表节点间直接跳转、条件跳转以及函数调用关系。其中，直接跳转和条件跳转分别将JUMP和JUMPI所在的块视为边的起点，并将跳转地址对应的节点视为边的终点，如图4所示，节点block1与节点block2存在跨函数调用关系，边的类型为FUNCCALL。

为了实现对漏洞语义的高效抽取，同时提升图神经网络对漏洞特征的学习能力，本文提出基于关键指令构建智能合约核心图。该方法由核心节点选取、函数映射和核心图构建3个步骤组成，如图5所示，各步骤的主要工作内容如下。

1) 核心节点选取：通过遍历CFG中的节点，标记包含关键指令的核心节点，并通过这些节点构

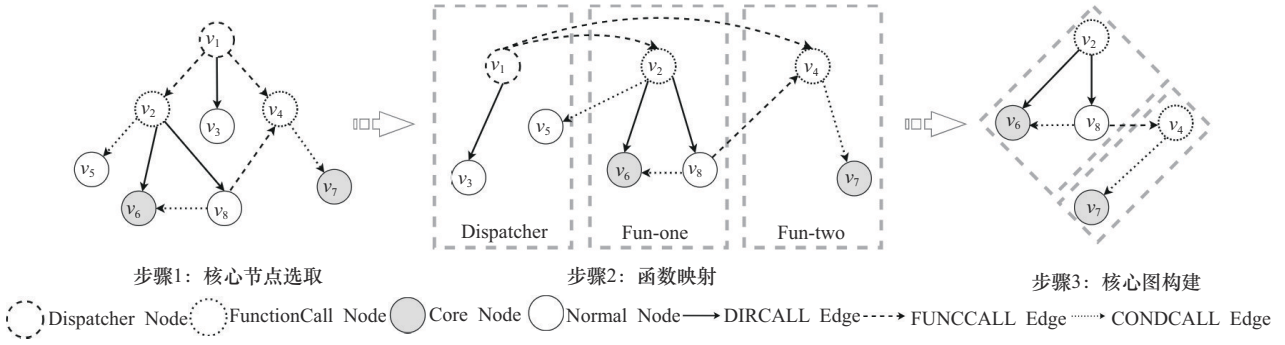


图5　智能合约核心图构建流程

建核心图骨架。以可重入漏洞为例，本文定义关键指令为CALL、MLOAD、SSTORE，其对应的核心节点为图5中v_6与v_7节点。

2) 函数映射：通过解析EVM指令序列，本文提出满足规定指令序列规则的节点为函数调用节点（图5中v_2与v_4节点）。分析函数调用节点中的起始偏移量、结束偏移量以及结束指令来计算下一节点跳转地址，从而完成函数内部节点的筛选，最终将CFG中的节点进行函数划分，并保留控制流依赖关系，完成函数映射。

3) 核心图构建：利用函数映射后的核心节点以及控制流关系进行核心图构造，因其只包含与漏洞关注点具有依赖关系的节点和边，故在保留CFG结构信息的同时，排除了与漏洞无关的信息。仅以单个函数为基础构建核心图会存在漏洞语义的缺失，因此本文的构建方法实现了跨函数的节点提取与相应的逻辑边构造，最终形成了以合约漏洞信息为核心的智能合约核心图，如图5所示。

2.3　预训练模型

为了得到更为准确的节点语义表征，面向智能合约字节码的漏洞检测模型不仅要考虑EVM指令本身的语义表征，也要考虑节点中EVM指令间的关系特征。目前自然语言处理领域的预训练模型BERT[19]具备可捕捉节点的双向上下文信息的多头注意力机制，因此本文将预训练模型应用于智能合约的漏洞检测，将规范后的指令输入预训练任务掩码语言建模（MLM, masked language model）进行训练；通过预训练模型生成更为精准的EVM指令表征向量，从而提升下游漏洞检测任务的准确率。

具体而言，将核心图中的指令以节点为单位进行序列化作为模型的输入，给定一个节点的序列$V=[V_1,V_2,V_3,\cdots,V_n]$其中$n$是序列长度。将序列$V$中的每一个元素进行词嵌入表示，将其转换为能够被神经网络处理的向量$\boldsymbol{Y}^0$，对于V中的任意一个元素V_i，V_i的向量表示由指令嵌入向量$\boldsymbol{E}_{T_i}$和位置编码向量$\boldsymbol{P}_{T_i}$构成。BERT模型将向量$\boldsymbol{Y}^0$作为输入，利用多层注意力机制，通过N层Transformer来获取向量$\boldsymbol{Y}^0$的注意力关系表示。

为了增强漏洞语义与核心图之间依赖关系，本文通过预训练任务MLM学习EVM指令间关系特征。选取输入序列中15%的token，将其中80%替换为“[MASK]”，10%随机替换为其他token，剩下10%保持不变。对被[MASK]标签掩盖或替换掉的第i个词进行预测，得到第$\widehat{s_i}$的概率方程如式(1)所示。

$$p\left(\widehat{s_i}=j|S_m\right)=\frac{\exp\left(w_i\boldsymbol{f}\left(\boldsymbol{S}_m\right)_i\right)}{\sum_{n=1}^{N}\exp\left(w_n\boldsymbol{f}\left(\boldsymbol{S}_m\right)_i\right)} \tag{1}$$

其中，j为预测得到的指令，$\boldsymbol{f}\left(\boldsymbol{S}_m\right)_i$为MLM对被替换位置的预测输出，$w_i$是模型结构中标志位$i$的权重，$n$是$s_i$的可能标签数量。将被替换的指令元素集合表示为$M(S)$，此时的交叉熵损失函数如式(2)所示。

$$L_{\mathrm{MLM}}=-\sum_{\boldsymbol{s}_i\in M(\boldsymbol{S})}\log p\left(\widehat{\boldsymbol{s}_i}|\boldsymbol{S}\right) \tag{2}$$

经过预训练模型多层注意力机制后，节点序列V的指令得到了高效表征，并融合了指令间的关系特征，该向量应用于后续下游任务具有良好的效果。

2.4　基于事件驱动的时序图神经网络

本文提出ETGN模型进行智能合约漏洞检测，该模型提取合约图的语义信息以及时序信息，将传统的图神经网络进行事件建模，并结合自注意力机制有效提升时序事件执行的并发性，从而提取合约图特征进行高效的漏洞检测。基于事件驱动的时序图神经网络模型架构如图6所示，首先将核心图映

射进行节点与边的编码，基于节点执行顺序提取时序信息，并将其与节点编码进行连接来构造事件，随后利用自注意力机制对时序事件进行信息交互，提升事件执行的并发能力，最终使用平均池化进行全局向量表征，进行向量映射并产生漏洞标签。

ETGN 模型以核心图作为输入，采用图神经网络（GNN, graph neural network）学习图结构，生成节点以及边的向量表征。具体而言，通过多层感知机（MLP, multilayer perceptron）将节点和边特征映射到初始节点和初始边向量。

$$\boldsymbol{h}_i^{(l)} = \mathrm{MLP}\left(\boldsymbol{x}_i\right) \tag{3}$$

$$\boldsymbol{h}_{ij} = \mathrm{MLP}\left(\boldsymbol{x}_{ij}\right) \tag{4}$$

其中，$\boldsymbol{x}_i$表示节点v_i的原始特征，而$\boldsymbol{x}_{ij}$表示从节点v_i到v_j的边特征。节点在l时刻的低维编码表示为$\boldsymbol{h}_i^{(l)}$，连接每对节点的边编码为$\boldsymbol{h}_{ij}$。

将合约图建模为时间戳序列的事件，每个事件包含 4 个部分：原始节点v_i、目标节点v_j、边类型$\boldsymbol{e}_{ij}$、时序信息$\boldsymbol{t}_{ij}$。对于时序信息，本文根据图中节点执行的顺序来定义时序信息，并利用原始节点和目标节点的特征通过线性函数和周期性非线性函数进行计算，计算式为

$$\boldsymbol{h}_{t_{ij}} = \mathrm{T2V}\left(\boldsymbol{t}_{ij}\right) = \begin{cases} \omega_f \boldsymbol{t}_{ij} + \varphi_f f = 0 \\ \sin\left(\omega_f \boldsymbol{t}_{ij} + \varphi_f\right), 1 < f < F \end{cases} \tag{5}$$

其中，F是时间向量的维度，f是第f个维度。本文采用一个经典时间矢量编码方法，表示为 T2V。T2V 将时间向量的第一维度作为线性时间表示（$f = 0$），而将其他维度作为周期性非线性表示（$1 < f < F$）。此外，ω_f和φ_f是可学习的线性参数。由于执行指令的时间顺序在表征二进制合约图中起着重要的作用，因此给定事件类型$\boldsymbol{e}_{ij}$和节点信息v_i和v_j以及时序信息$\boldsymbol{t}_{ij}$，通过以下 4 个元素连接来对事件编码进行建模。在连续生成事件嵌入之后，根据时间顺序获得一系列带时间戳的事件。

$$\boldsymbol{e}_u = \left(\boldsymbol{h}_i^{(n)} \| \boldsymbol{h}_j^{(n)} \middle\| \boldsymbol{h}_{e_{ij}} \middle\| \boldsymbol{h}_{t_{ij}}\right) \tag{6}$$

$$\boldsymbol{E} = \left(\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_u\right) \tag{7}$$

由于大多数神经网络对时序信息的表示存在较大局限性，基于事件的自注意力机制将提升事件执行的并发性。本文将合约图表示为事件序列，并与自注意力层相结合进行事件编码，自注意力层处理输入事件如式(8)和式(9)所示。

$$\hat{\boldsymbol{E}} = \mathrm{Attention}\left(\boldsymbol{Q}_e, \boldsymbol{K}_e, \boldsymbol{V}_e\right) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}_e \boldsymbol{K}_e^{\mathrm{T}}}{\sqrt{\boldsymbol{D}}}\right)\boldsymbol{V}_e \tag{8}$$

$$\hat{\boldsymbol{g}} = \mathrm{Avgpooling}\left(\sum_{i=0}^{U} \widehat{\boldsymbol{e}}_i\right) \tag{9}$$

其中，$\boldsymbol{D}$是$\boldsymbol{e}$的维度，$\boldsymbol{Q}_e$、$\boldsymbol{K}_e$和$\boldsymbol{V}_e$分别表示每个事件的键、查询和值的参数，本文聚合所有基于时序的事件$\widehat{\boldsymbol{e}}_i$并使用全局平均池化层来计算最终的图特征$\hat{\boldsymbol{g}}$。ETGN 模型在漏洞判定阶段，将图特征$\hat{\boldsymbol{g}}$进行向量映射，产生漏洞标签 0/1，1 代表合约存在漏洞，0 代表该合约无漏洞。

3　实验设计

3.1　实验设置

本文采用以太坊平台上真实的智能合约，数据集包含 120 932 份以太坊智能合约，针对可重入漏洞、时间戳依赖漏洞，以及 Tx.origin 身份认证漏洞，在源码中必须存在相应的关键字才有几率触发漏洞。为了验证本文方法的有效性，各种类型漏洞的合约均包含可触发漏洞的关键字，其中 20 013 份合约的源码含有能够触发可重入漏洞的 call.value 关键字；67 833 份合约源码中有 block.timestamp 语句，可能导致时间戳依赖漏洞；21 028 份合约中至

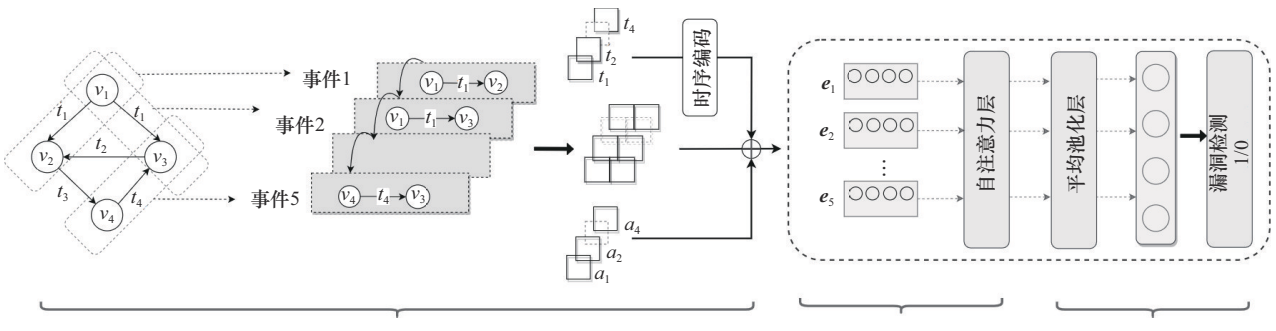


图6　基于事件驱动的时序图神经网络模型架构

少包含一个调用Tx.origin的语句，使得它们可能受到身份认证攻击的影响，具体数据情况如表3所示。此外，针对模型的可泛化性研究，本文额外构建了60 178份智能合约，包含蜜罐漏洞、拒绝服务漏洞、整数溢出等8种类型。

为了验证核心图和预训练模块效果，本文将未经核心图构建的合约图（保留全部节点）定义为合约原始图（primal graph），经过核心图构建而生成的合约图定义为关键图（core graph）。在本文中Primal G-W代表针对原始图数据使用Word2vec进行编码，Core G-W与Core G-B分别代表针对关键图层面使用Word2vec与BERT进行编码。

本文使用准确率（Accuracy）、精确率（Precision）、召回率（Recall）和F1值（F1 Score）作为评价指标。

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{10}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

3.2 实验结果与分析

本文所设计的实验用于回答以下3个研究问题。

RQ1：本文方法进行漏洞检测是否优于已有的智能合约漏洞检测方法?

RQ2：本文提出的核心图以及预训练是否提升漏洞检测准确度?

RQ3：本文提出的ETGN模型是否优于现有神经网络漏洞检测模型，其事件生成模块对检测效率提升效果如何?

3.2.1 智能合约漏洞检测效果对比(RQ1)

为了验证漏洞检测效果，将本文提出的方法与7种现有检测方法在以太坊数据集上进行实验对比，包括4种传统检测方法（Oyente、 Mythril、Smartcheck、Securify）以及4种基于深度学习的最新方法。基于深度学习的智能合约漏洞自动检测（VSCL, automating vulnerability detection in smart contracts with deep learning）、EtherGIS、消息传递神经网络（MPNN, message passing neural network）、时序消息传递（TMP, temporal message propagation）网络。其中，VSCL[14]在CFG上使用深度优先遍历得到一个新的具有智能合约执行顺序信息的操作码序列，然后将操作码序列转换为向量表示，将向量输入到深度神经网络（DNN, deep neural network）模型中进行训练和分类。EtherGIS[20]是利用图神经网络和专家知识从CFG中提取图特征的漏洞检测框架，构建敏感EVM指令语料库，根据语料库确定CFG中节点的特征，最终生成相应的属性图。MPNN是消息传递神经网络，通过传递和聚合消息来捕捉CFG中节点之间的关系，最终节点可以获取整个CFG的信息。TMP[21]包括图生成、图归一化和图神经网络训练3个部分，其提出了一种基于时间次序的时序图神经网络模型并应用于合约漏洞检测。表4列出了所有方法在不同智能合约漏洞类型上的检测效果。

在可重入漏洞检测方面，传统的Oyente、Smartcheck、Securify等工具各项指标均较低，例如传统工具Securify的准确率为71.89%，最新神经网络检测方法VSCL、EtherGIS、TMP分别比其高出12.79%、6.48%、13.69%，而本文方法比Securify高出22.78%，比最新方法VSCL、EtherGIS、TMP分别高出9.99%、16.30%、9.09%。该结果充分说明了预训练技术与基于事件的时序图神经网络的先进性。进一步分析，首先注意到MPNN的准确率仅有42.56%，原因是本文针对MPNN的输入并无任何图规范操作，将整个CFG信息输入神经网络进行漏洞检测，大量的噪声节点影响神经网络对漏洞语义的学习。此外，可重入漏洞原理相比其他两种漏洞类型逻辑更为复杂，因此导致MPNN的准确率较低。其次，相对于本文采用的核心图构建方法，VSCL对CFG采用深度优先遍

表3 实验数据集

数据项	合约数量/份	漏洞数量/个	描述
合约信息	120 932	29 783	源自以太坊的智能合约
含call.value关键字	20 013	4 693	可能受到可重入性漏洞的影响
含block.timestamp关键字	67 833	19 836	可能导致时间戳依赖性漏洞
含origin关键字	21 028	5 254	可能受到身份认证攻击

表 4　　实验结果对比

检测方法	可重入漏洞				时间戳依赖漏洞				Tx.origin 身份认证漏洞			
	准确率	召回率	精确率	F1 Score	准确率	召回率	精确率	F1 Score	准确率	召回率	精确率	F1 Score
Oyente	61.62%	54.71%	38.16%	44.96%	59.45%	38.44%	45.16%	41.53%	—	—	—	—
Mythril	60.54%	71.69%	39.58%	51.02%	61.08%	41.72%	50.00%	45.49%	—	—	—	—
Smartcheck	52.97%	32.08%	25.00%	53.57%	44.32%	37.25%	39.16%	38.18%	—	—	—	—
Securify	71.89%	56.60%	50.85%	53.57%	—	—	—	—	—	—	—	—
DNN (Primal G-W)	54.95%	72.91%	39.47%	51.21%	62.73%	71.42%	47.61%	57.14%	67.76%	76.08%	51.47%	61.40%
MPNN (Primal G-W)	42.56%	40.27%	25.55%	31.26%	73.50%	62.30%	58.69%	60.44%	54.94%	54.34%	38.16%	44.84%
GCN (Primal G-W)	59.90%	20.13%	31.52%	24.57%	55.90%	49.10%	39.28%	43.65%	64.10%	80.43%	48.05%	60.16%
DNN (Core G-W)	86.71%	70.86%	87.70%	78.38%	84.78%	75.96%	76.69%	76.32%	86.02%	79.80%	77.57%	78.67%
MPNN (Core G-W)	84.45%	73.50%	79.28%	76.28%	88.19%	86.53%	78.94%	82.56%	80.42%	72.94%	65.95%	69.27%
GCN (Core G-W)	84.23%	72.84%	79.13%	75.86%	86.02%	77.88%	78.64%	78.26%	82.91%	64.42%	78.82%	70.88%
DNN (Core G-B)	88.06%	90.06%	78.16%	83.69%	89.75%	94.23%	78.40%	85.58%	90.74%	95.29%	78.64%	86.17%
MPNN (Core G-B)	88.96%	94.03%	78.02%	85.28%	90.06%	96.15%	78.12%	86.20%	91.45%	95.29%	80.19%	87.09%
GCN (Core G-B)	88.96%	76.15%	89.84%	82.43%	89.44%	89.42%	80.17%	84.54%	89.32%	84.70%	80.89%	82.75%
VSCL	84.68%	60.26%	91.91%	72.79%	79.81%	51.92%	78.26%	62.42%	85.76%	60.00%	89.47%	71.83%
EtherGIS	78.37%	54.30%	75.22%	63.07%	77.92%	59.02%	68.53%	63.43%	70.49%	59.02%	54.14%	56.47%
TMP	85.58%	71.05%	83.72%	76.86%	85.30%	62.74%	83.11%	71.50%	83.18%	69.58%	77.43%	73.30%
Core-BERT-ETGN	94.67%	96.71%	88.55%	92.45%	93.08%	98.02%	81.81%	89.19%	95.00%	95.74%	90.00%	92.78%

历去掉大量无关节点；EtherGIS 则构建敏感 EVM 指令语料库，对关键指令有较高敏感性；而 TMP 对合约图进行了归一化，聚焦于核心语义。实验结果充分说明本文提出的核心图构建可以将合约漏洞语义进行有效表征，且对下游漏洞检测任务有良好的效果。

在时间戳漏洞检测方面，传统工具 Mythril 准确率为 61.08%，这是因为现有方法通过粗粒度地检查函数中是否存在 block.timestamp 语句来检测时间戳依赖漏洞，从而导致准确率较低；Smartcheck 根本上依赖于一些简单的逻辑规则来检测漏洞，这导致准确率和 F1 Score 较低；本文方法在时间戳依赖漏洞检测方面 4 个指标保持最佳性能，准确率分别比 VSCL、EtherGIS、MPNN、TMP 高出 13.27%、15.16%、19.58%、7.78%。由于时间戳依赖漏洞相比于其他类型漏洞关键指令较多，所以对时间戳依赖漏洞的检测更注重合约整体语义的捕获。实验结果表明，本文预训练模型 BERT 充分考虑指令语义关系表征，提高节点向量表征精度，且 ETGN 模型能够有效捕捉节点的控制流依赖关系和时序信息，从而实现高效的漏洞检测。

在 Tx.origin 身份认证漏洞检测方面，本文方法与 VSCL、EtherGIS、MPNN、TMP 进行对比，本文方法准确率达到了 95%，比其他方法分别高出了 9.24%、24.51%、40.05%、11.82%。分析 Tx.origin 身份认证漏洞原理可知，指令中存在 ORIGIN、MLOAD 则较大概率存在漏洞。MPNN 由于

全图输入神经网络，无关节点的特征过多导致无法进行语义聚焦，准确率较低；EtherGIS构建了敏感EVM指令库，针对Tx.origin身份认证漏洞有较高的敏感性；VSCL利用关键路径过滤了大量节点，其模型逐层预训练可以提升特征学习效率，所以在Tx.origin身份认证漏洞有不错的表现；TMP图生成模块将图进行消融，保留了合约图的核心语义特征，所以在Tx.origin身份认证漏洞上达到了较高的检测准确率；本文基于关键指令构建核心图的方式，将ORIGIN、MLOAD作为核心节点，经过预训练模型的高度表征后，漏洞检测效率更高。

3.2.2 核心图以及预训练效果（RQ2）

为了评估核心图与预训练模型BERT对漏洞检测能力的贡献，本文分别在原始图、关键图以及预训练模型后的表征层面进行对比实验，并通过现有神经网络方法DNN、GCN、MPNN进行验证。

首先，本文将原始图输入神经网络中，在可重入漏洞方面，准确率最高的传统工具Securify相比上述3种传统神经网络分别高出了16.94%、11.99%、29.33%。由于可重入漏洞逻辑较为复杂，因此传统神经网络的表现不如工具检测；在时间戳依赖漏洞检测方面，Mythril准确率比GCN高出5.18%，但DNN与MPNN分别比Mythril高出1.65%、12.42%。进一步分析，时间戳依赖漏洞注重合约图语义，因此传统神经网络效果优于工具检测。上述实验结果证明，利用原始图作为输入效果较差，原始图中存在的噪声节点以及对关键节点定义缺失都会影响漏洞语义的检测。

其次，本文使用核心图构建方法生成关键图并利用Word2vec进行表征，将其传入神经网络进行检测。传统神经网络在可重入漏洞的检测效果相比于全图提升24.33%~41.89%；时间戳依赖漏洞提升14.69%~30.12%；Tx.origin身份认证漏洞检测效果提升18.26%~25.48%。由于VSCL中利用了DNN进行训练，本文额外使用基于核心图的DNN方法与VSCL方法进行对比，上述3种漏洞准确率分别提升了2.03%、4.97%与0.26%，召回率分别提升了10.6%、24.04%与19.80%。同理，EtherGIS利用了GCN进行分类，本文使用基于核心图的GCN方法与EtherGIS方法进行对比，准确率平均提升了8.79%，召回率平均提升了14.26%。综上所述，本文提出基于关键指令构建核心图方法极大提高了漏洞检测的准确率和召回率，即使将传统神经网络与核心图技术结合也会表现出较高的检测能力。

最后，在关键图的基础上，本文通过预训练模型代替Word2vec进行向量表征，进一步检测预训练技术在智能合约漏洞检测上产生的效果。结果显示，基于核心图结合预训练模型的方式可以继续提升模型性能，在可重入漏洞方面MPNN、GCN、DNN准确率分别提升了4.51%、4.73%、1.35%。其中MPNN与DNN的召回率在3个漏洞方面都达到了90%以上。因此，上述实验结果验证了预训练模型BERT对漏洞语义能够实现更准确的表征，也为下游漏洞检测任务提供了良好的检测效果。

3.2.3 ETGN模型效果（RQ3）

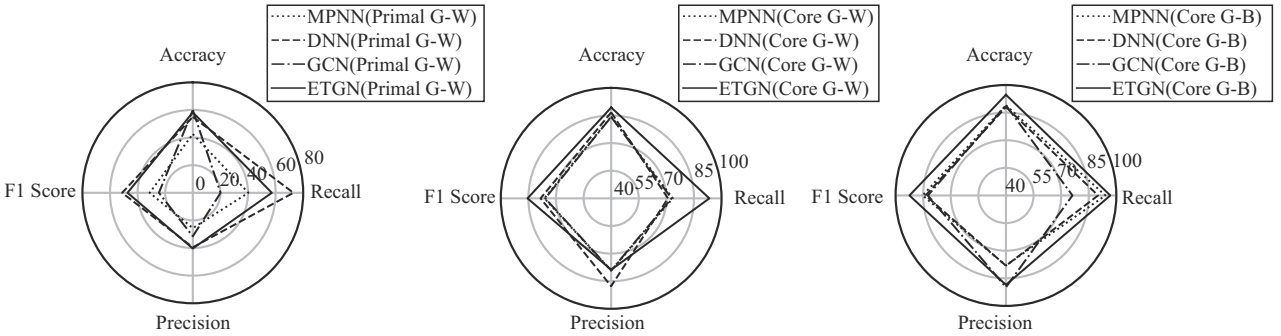
1) 与现有神经网络模型对比

为了验证ETGN模型的有效性，本文将传统神经网络DNN、GCN、MPNN替换ETGN模型进行结果对比。本文提出的核心图构建与BERT预训练技术也将应用于传统神经网络，分别在原始图和关键图2个层面上进行对比，ETGN模型漏洞检测结果的可视化效果如图7所示：每个雷达的上下左右4个部分，分别代表准确率、精确率、F1值、召回率。雷达线条越向外扩散，代表该性能值越高。

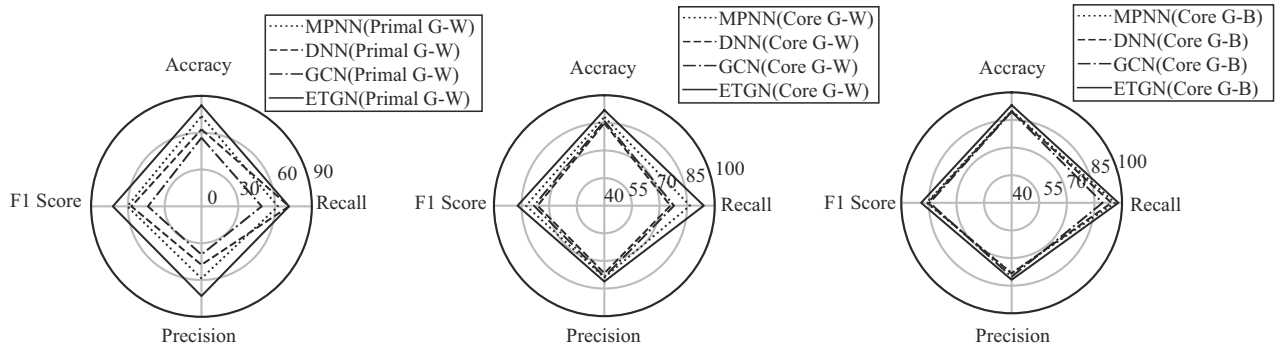
实验结果表明，无论是利用Word2vec还是BERT作为向量表征，ETGN模型在关键图上的表现均优于传统神经网络，特别是利用核心图与预训练技术相结合的方式，使得ETGN模型在3个类型漏洞上准确率都达到93%以上，而召回率全部在95%以上。然而，ETGN模型在原始图层面表现较差，这是因为原始图存在过多噪声节点，ETGN模型考虑节点、边以及其时序信息，因此对模型的漏洞检测会产生很大的干扰，从而降低检测精度。通过核心图对节点的自动筛选，节点和逻辑边的数量大幅度减少，使得ETGN模型能够充分学习关键图中信息从而进行高精度的漏洞检测。

2) 事件生成模块的有效性验证

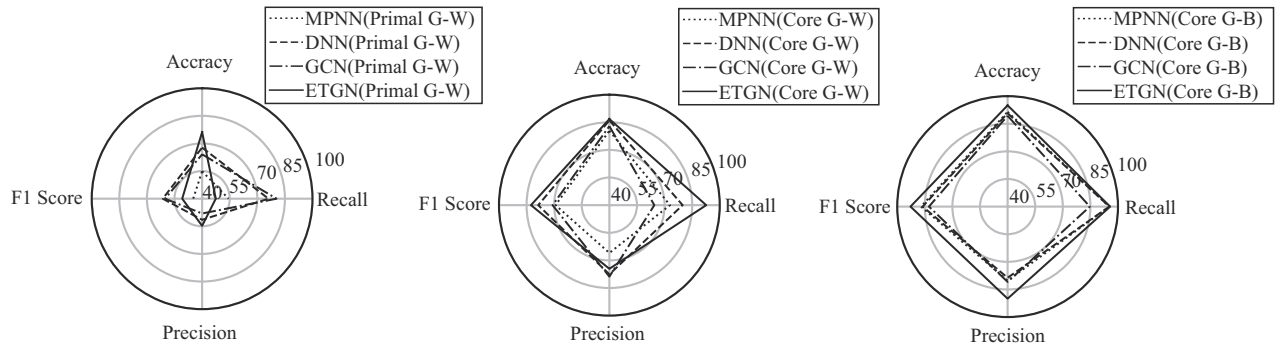
为了验证ETGN模型中事件生成模块（EGM, event generation module）的确提升了漏洞检测准确度，本文对EGM展开消融实验。将ETGN模型进行保留EGM以及去除EGM实验对比，实验结果如表5和表6所示。



(a) 可重入漏洞



(b) 时间戳依赖漏洞



(c) Tx.origin身份认证漏洞

图7　ETGN 模型漏洞检测结果的可视化效果

表5　　ETGN(去除 EGM)实验结果

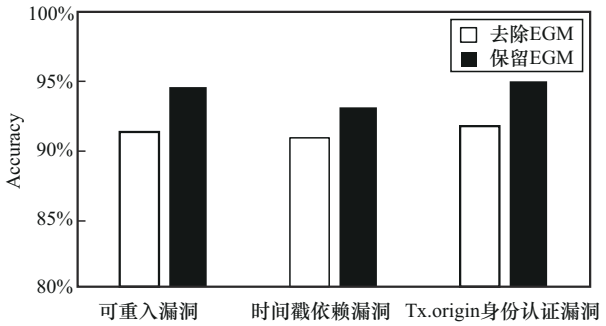
检测方法	可重入漏洞				时间戳依赖漏洞				Tx.origin 身份认证漏洞			
	准确率	召回率	精确率	F1 Score	准确率	召回率	精确率	F1 Score	准确率	召回率	精确率	F1 Score
Primal G-W	58.25%	57.25%	40.32%	47.31%	82.28%	71.42%	73.07%	72.24%	76.20%	47.42%	54.76%	50.82%
Core G-W	89.35%	93.37%	78.77%	85.45%	91.93%	94.05%	81.19%	87.15%	86.78%	92.63%	74.57%	82.62%
Core G-B	**94.67%**	**96.71%**	**88.55%**	**92.45%**	**93.08%**	**98.02%**	**81.81%**	**89.19%**	**95.00%**	**95.74%**	**90.00%**	**92.78%**

表6　　ETGN(保留 EGM)实验结果

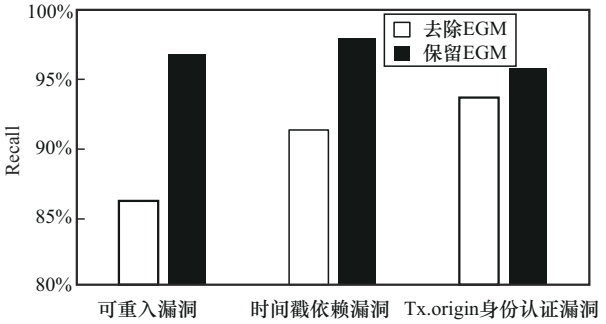
检测方法	可重入漏洞				时间戳依赖漏洞				Tx.origin 身份认证漏洞			
	准确率	召回率	精确率	F1 Score	准确率	召回率	精确率	F1 Score	准确率	召回率	精确率	F1 Score
Primal G-W	57.25%	22.30%	29.29%	25.32%	75.00%	72.93%	59.14%	65.31%	76.21%	44.77%	71.42%	55.04%
Core G-W	82.70%	84.76%	69.78%	76.64%	87.60%	75.49%	81.05%	78.17%	83.59%	79.47%	73.61%	76.43%
Core G-B	**91.35%**	**86.09%**	**87.83%**	**86.95%**	**90.77%**	**91.17%**	**80.17%**	**85.32%**	**91.78%**	**93.61%**	**83.80%**	**88.44%**

本文首先考虑在关键图层面使用Word2vec进行编码，分析可重入漏洞、时间戳依赖漏洞以及Tx.origin身份认证漏洞。保留EGM的ETGN模型准确率分别提升了6.65%、4.33%、3.19%，召回率分别提升了8.61%、18.56%、13.16%。经过对比，保留EGM可以提升ETGN模型漏洞检测的准确率，并极大提升了召回率，有效说明EGM将漏洞语义精准表征，促进漏洞检测。

为了进一步验证EGM对漏洞语义的表征能力，本文在核心图与预训练技术基础上将ETGN模型进行消融实验对比。根据图8漏洞检测的对比结果，本文发现保留EGM的模型其准确率和召回率在3个漏洞层面效果皆优于去除EGM的ETGN模型。其中，准确率高出2.31%~3.32%，召回率高出2.13%~10.62%，验证了EGM对漏洞语义的高表征能力。



(a) 准确率对比



(b) 召回率对比

图8　消融实验对比

根据上述分析，EGM建模包含了事件类型、节点信息以及时序信息，将图转化为一系列带有时间戳的事件，结合ETGN模型的自注意力机制提高了事件执行的并发性，能够将时序信息融入漏洞检测之中，从而提升了漏洞检测的能力。

3) 模型泛化能力验证

本文方法在现有可重入漏洞、时间戳依赖漏洞以及Tx.origin身份认证漏洞上均有着出色的表现。为了验证模型的泛化能力，本文在原有12万份智能合约基础上新增了60 178份智能合约，其中包括蜜罐漏洞、拒绝服务漏洞、整数溢出等8种类型漏洞样本进行实验。新增漏洞实验结果对比如表7所示。

表7　　新增漏洞实验结果对比

漏洞类型	准确率	召回率	精确率	F1 Score
蜜罐漏洞	90.76%	88.23%	78.94%	71.42%
构造函数名称错误漏洞	95.61%	84.96%	90.76%	97.41%
强制馈送漏洞	91.26%	77.06%	79.62%	82.35%
拒绝服务漏洞	82.57%	72.99%	68.49%	64.51%
调用接口错误漏洞	89.32%	81.02%	79.56%	78.16%
整数溢出漏洞	82.17%	73.55%	66.41%	60.54%
条件竞争漏洞	93.68%	80.15%	86.32%	93.51%
未检查call返回值漏洞	84.50%	80.340%	70.94%	63.51%

由表7可知，本文方法漏洞检测的准确率均达到80%以上，其中蜜罐漏洞、构造函数名称错误漏洞、强制馈送漏洞和条件竞争漏洞准确率达到了90%。上述结果充分说明了ETGN模型具有优秀的泛化能力。通过EGM建模，ETGN模型有效捕捉了合约图控制流依赖关系以及节点跳转的时序信息，高效表征漏洞语义信息。此外，将事件生成结合自注意力机制，极大地提升了事件执行的并发能力，使得ETGN模型具有更好的漏洞检测效率。

4　结束语

本文提出了一种基于预训练与时序图神经网络的智能合约漏洞检测方法。为了充分考虑合约字节码的特征和漏洞语义，将字节码转换成CFG，并根据关键指令构建核心图；为了高效表征核心图语义信息，利用预训练模型BERT对核心图中节点信息进行编码；最后使用基于事件驱动的时序图神经网络，将核心图中漏洞语义信息进行事件建模，实现高精度的漏洞检测。在3个漏洞类型数据集上的实验结果表明，本文所提出方法的漏洞检测性能优于其他现有方法。另外，额外增加60 178份智能合约共8种类型漏洞进行实验验证，漏洞检测准确率均达到80%以上，充分证明了本文方法的可泛化性。

在未来的研究中，将尝试设计新型神经网络以期自主学习合约中的漏洞特征，同时尝试更高效的字节码表示结构和预训练模型，进一步提高准确率并降低误报率，提升字节码层面漏洞检测方法的性能。

参考文献:

[1] 王利朋, 关志, 李青山, 等. 区块链数据安全服务综述[J]. 软件学报, 2023, 34(1): 1-32.
WANG L P, GUAN Z, LI Q S, et al. Survey on blockchain-based security services[J]. Journal of Software, 2023, 34(1): 1-32.

[2] 陈锦富, 王震鑫, 蔡赛华, 等. 基于蜕变测试的区块链智能合约漏洞检测方法[J]. 通信学报, 2023, 44(10): 164-176.
CHEN J F, WANG Z X, CAI S H, et al. Vulnerability detection method for blockchain smart contracts based on metamorphic testing[J]. Journal on Communications, 2023, 44(10): 164-176.

[3] 吴恺东,马郓,蔡华谦,等. 面向智能合约分片的联盟区块链系统[J]. 软件学报,2023, 34(11): 5042-5057.
WU K D, MA Y, CAI H Q, et al. Consortium blockchain system based on smart contract-oriented sharding[J]. Journal of Software, 2023, 34 (11): 5042-5057.

[4] BADRUDDOJA S, DANTU R, HE Y Y, et al. Making smart contracts smarter[C]//Proceedings of the 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). Piscataway: IEEE Press, 2021: 1-3.

[5] 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). Piscataway: IEEE Press, 2019: 8-15.

[6] JIANG B, LIU Y, CHAN W K. ContractFuzzer: fuzzing smart contracts for vulnerability detection[C]//Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. New York: ACM Press, 2018: 259-269.

[7] 王友卫, 侯玉栋, 凤丽洲. 基于源码结构和图注意力网络的以太坊蜜罐合约检测方法[J]. 通信学报, 2023, 44(9): 161-172.
WANG Y W, HOU Y D, FENG L Z. Honeypot contract detection method for Ethereum based on source code structure and graph attention network[J]. Journal on Communications, 2023, 44(9): 161-172.

[8] MUELLER B,HONIG J,PARASARAM N, et al. Mythril-reversing and bug hunting framework for the Ethereum blockchain[R]. 2017.

[9] TSANKOV P, DAN A, DRACHSLER-COHEN D, et al. Securify: practical security analysis of smart contracts[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2018: 67-82.

[10] KALRA S, GOEL S, DHAWAN M, et al. ZEUS: analyzing safety of smart contracts[C]//Proceedings of the 2018 Network and Distributed System Security Symposium. Reston: Internet Society, 2018: 26-35.

[11] MA F C, XU Z Y, REN M, et al. Pluto: exposing vulnerabilities in inter-contract scenarios[J]. IEEE Transactions on Software Engineering, 2022, 48(11): 4380-4396.

[12] CAMINO R, TORRES C F, BADEN M, et al. A data science approach for detecting honeypots in ethereum[C]//Proceedings of the 2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). Piscataway: IEEE Press, 2020: 1-9.

[13] 张红霞, 王琪, 王登岳, 等. 基于深度学习的区块链蜜罐陷阱合约检测[J]. 通信学报, 2022, 43(1): 194-202.
ZHANG H X, WANG Q, WANG D Y, et al. Honeypot contract detection of blockchain based on deep learning[J]. Journal on Communications, 2022, 43(1): 194-202.

[14] MI F, WANG Z Y, ZHAO C, et al. VSCL: automating vulnerability detection in smart contracts with deep learning[C]//Proceedings of the 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). Piscataway: IEEE Press, 2021: 1-9.

[15] ZHANG L J, WANG J L, WANG W Z, et al. A novel smart contract vulnerability detection method based on information graph and ensemble learning[J]. Sensors, 2022, 22(9): 3581.

[16] HUANG J J, HAN S M, YOU W, et al. Hunting vulnerable smart contracts via graph embedding based bytecode matching[J]. IEEE Transactions on Information Forensics and Security, 2021, 16: 2144-2156.

[17] FAN Y Q, SHANG S Y, DING X. Smart contract vulnerability detection based on dual attention graph convolutional network[C]//Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing. Berlin: Springer, 2021: 335-351.

[18] ANGELO M D, SALZER G. A survey of tools for analyzing ethereum smart contracts[C]//Proceedings of the 2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON). Piscataway: IEEE Press, 2019: 69-78.

[19] DEVLIN J, CHANG M W, LEE K, et al. BERT: pre-training of deep bidirectional transformers for language understanding[J]. arXiv Preprint, arXiv: 1810.04805v2, 2018.

[20] ZENG Q R, HE J H, ZHAO G S, et al. EtherGIS: a vulnerability detection framework for ethereum smart contracts based on graph learning features[C]//Proceedings of the 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC). Piscataway: IEEE Press, 2022: 1742-1749.

[21] ZHUANG Y, LIU Z G, QIAN P, et al. Smart contract vulnerability detection using graph neural network[C]//Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. California: International Joint Conferences on Artificial Intelligence Organization, 2020: 3283.

[作者简介]

庄园（1988– ），女，黑龙江哈尔滨人，博士，哈尔滨工程大学副教授、硕士生导师，主要研究方向为区块链安全、人工智能和高性能计算。

樊泽楷（1998– ），男，山西晋城人，哈尔滨工程大学硕士生，主要研究方向为区块链安全、人工智能。

孙建国（1981– ），男，黑龙江哈尔滨人，博士，西安电子科技大学教授、博士生导师，主要研究方向为数据安全、人工智能和工业互联网。

王诚（2001– ），男，河北定州人，哈尔滨工程大学硕士生，主要研究方向为网络安全、人工智能。

李耀麟（1993– ），男，河北邢台人，北京理工大学博士生，主要研究方向为自然语言处理、区块链安全。