

文章编号: 2096-1618(2025)01-0007-07

异构图的智能合约漏洞检测方法

侯羿杉, 王 焱

(成都信息工程大学网络空间安全学院, 四川 成都 610225)

摘要:针对现有的基于深度学习智能合约漏洞检测方法无法有效利用上下文信息,提出一种基于异构图的智能合约漏洞检测方法。通过将合约源码解析为包含数据流和控制流的符号图,然后使用图神经网络对图进行表征学习,并通过神经网络进行漏洞预测。在 ESC 和 VSC 两个数据集上进行实验,和现有工具以及模型进行对比,结果表明该方法在准确率、召回率、精度、F1 分数 4 个指标均取得提升。

关键词:智能合约;漏洞检测;深度学习;图神经网络

中图分类号:TP183

文献标志码:A

doi:10.16836/j.cnki.jcuit.2025.01.002

0 引言

2008 年,中本聪发布了比特币白皮书^[1],这标志着区块链技术^[2]的兴起。2009 年,比特币正式发行,展现出区块链技术的广阔应用前景,吸引了学术界和业界的广泛关注,开启了区块链1.0的时代。随着对区块链技术的深入研究,发现区块链不仅仅局限于去中心化的货币交易中,智能合约的出现加速了区块链技术在加密货币之外的其他领域的应用。以太坊是第一个支持开发智能合约的区块链平台,它的发布标志着区块链2.0时代的开始。通过区块链的去中心化共识机制,智能合约允许不相互信任的用户在无需第三方可信机构的情况下完成数据交换或交易,这使智能合约成为区块链最具应用价值的领域,随后涌现了越来越多支持智能合约的区块链平台。目前,以太坊是最广泛使用的支持智能合约的区块链平台,上面存在超过 5000 万份智能合约。以太坊本身的市值也超过 1500 亿美元。然而,作为在区块链中运行的程序,智能合约控制着大量资金,不可避免地存在由程序缺陷导致的安全漏洞^[3-4]。因此,如何准确及时地检测各种智能合约的漏洞成为区块链安全研究的重点和热点。区块链技术的发展进一步促进智能合约在实际应用中的探索和改进,为区块链技术的未来发展打下坚实的基础。

为确保智能合约的安全性,研究人员积极研究智能合约漏洞检测技术,并开发了多种工具来检测和预防现有的智能合约漏洞^[5]。目前,主要采用的技术包

括符号执行^[6-7]、形式化验证^[8-9]和模糊测试^[10-11]。符号执行是一种广泛应用的方法,它将不确定的输入转化为符号值,并利用约束求解器对程序执行路径进行求解。这种方法能够实现更准确、更全面的程序分析,但常常面临路径爆炸等问题。

形式化验证主要采用严格的可演绎描述语言或逻辑来描述程序的属性和特征,并使用数学逻辑证明和推理来构建形式化规范,以确定安全属性是否符合预期。然而,形式化验证方法需要较强的逻辑推理能力,且自动化程度相对较低。

模糊测试是另一种常用的方法,利用随机生成的测试样本作为智能合约的输入,并通过监控合约执行过程来判断是否触发程序漏洞或其他异常行为。虽然模糊测试方法能够有效检测合约漏洞,但需要提前获取智能合约的源代码和 ABI 接口信息。

近年来,基于深度学习的程序分析方法在安全检测领域逐渐流行。这种方法自动化程度较高,可以从大量数据中提取程序的隐藏特征,突破了基于规则的传统漏洞检测方法的局限性。深度学习方法在智能合约漏洞检测方面展现出了潜力,并在安全检测领域受到越来越多的关注。目前,常用的方法是将代码的不同表示形式(如抽象语法树、程序依赖图等)转换为一维序列,并利用深度学习模型(如卷积神经网络或循环神经网络)对代码进行表示学习。然而,因为程序中存在复杂的上下文关系,将程序表示为一维序列难以准确捕捉程序的结构和上下文信息。

鉴于已有的基于静态分析的漏洞检测方法存在准确率不高的问题,以及普通的图神经网络在传播信息时本质上是平坦的,忽略了不同节点的重要性是不相等的这一性质。本文提出一种基于异构图的智能合约

收稿日期:2023-09-15

基金项目:四川省科技计划资助项目(2023DYF0380、2021ZYD0011);
国家社会科学基金资助项目(23BSH061)

通信作者:王焱. Email:wangyi1177@cuit.edu.cn

漏洞检测方法,设计了数据流和控制流两种边类型的提取,以及变量节点、函数节点和 Fallback 节点的提取。并构建异构图神经网络模型,以异构图作为输入,学习不同类型的边以及节点对漏洞产生的影响,通过注意力机制进行节点级别和语义级别的聚合,最后得到图级的特征,对图进行分类判断是否存在漏洞。

1 相关工作

1.1 现有智能合约检测工具

为解决智能合约安全问题,已出现了许多漏洞检测工具。Oyente^[6]是最早的基于符号执行的漏洞检测工具;Securify^[12]是基于静态分析,通过字节码来检测智能合约漏洞的工具;SmartCheck^[13]通过 solidity 源码来进行分析的检测工具;Slither^[14]通过对编译得到 solidity 抽象语法树进行分析来检测漏洞。

此外,还有许多检测特定漏洞的工具。Jolt^[15]通过动态检测方式判断无限循环漏洞;PDA^[16]是一种基于符号执行的无限循环漏洞检测工具;SMT^[17]是一种基于可满足性模块理论的检测方法;Looper^[18]是一种对运行中程序进行动态分析的无限循环漏洞检测工具。

1.2 神经网络

神经网络是一种由多个神经元(或称为节点)组成的计算模型,通常分为输入层、隐藏层(可能有多层)、输出层。每个神经元与前后层的神经元相连接,通过权重来传递信息。

基于传统神经网络的检测方法分为文本处理和图像处理两类。基于文本处理的方法通常是将智能合约源码转化为字节码和操作码,通过自然语言处理的方式来提取特征,然后进行训练。基于图像处理的方法大多是将字节码和操作码转为 RGB 图像,通过图像处理的方式来提取特征。

1.3 图神经网络

图神经网络(graph neural network, GNN)作为一种专门用于处理图数据的深度学习模型,近年来取得了显著的进展,并在社交网络分析、推荐系统、生物信息学等领域取得了成果。与传统的神经网络不同,图神经网络能够直接对图数据中的节点和边进行学习,并捕捉节点之间的复杂关系。

图神经网络与传统神经网络有许多不同:(1)在样本处理上。神经网络只处理独立的样本,图神经网络可以在图级别和节点级别进行训练。图级别训练是将整个图作为输入进行训练,模型通过对整个图的特

征进行传播和聚合来预测。节点级别训练是将每个节点作为独立的样本进行训练,模型通过节点的邻居信息进行特征传播和预测。(2)在网络信息传递上。图神经网络通过迭代传播机制,在图中传递和更新节点的特征表示。这与传统神经网络中的前向传播过程不同,传统神经网络只进行一次前向传播,而图神经网络需要多次迭代传播,以便节点的特征能在图上进行有效的信息交流和更新。

图神经网络的优势在于对局部邻域信息的聚合和全局图结构的建模,这使它在处理智能合约中复杂的代码时具有巨大潜力。通过将智能合约转换成图数据表示,可以利用图神经网络来挖掘代码中的模式、异常行为以及潜在的漏洞。TMP^[19]是第一个基于图神经网络的智能合约漏洞检测模型。区别于传统的消息传递机制,它提出了一种基于时间序列沿着边传递消息的传播机制。AMP^[20]模型在 TMP 模型的基础上结合专家规则使检测效果进一步提高。

1.4 异构图

异构图^[21]是指节点类型的数量加上边类型的数量大于2,也就是包含多种类型节点和边的图。在异构图中提出了一种元路径^[21]用于提取语义信息。

元路径是一种用于描述图中不同类型节点之间关系的概念,异构图是由多种不同类型的节点和它们之间的边构成的图。如社交网络中的用户和商品,或者学术网络中的作者和论文。它可以看作一种特定的节点序列,其中节点类型和边类型交替出现,用来定义一种特定类型的关系路径。

在智能合约漏洞检测中,将智能合约源代码转换为异构图可以更好地捕捉合约中不同类型代码块之间的关联关系和依赖关系。在智能合约中,可能存在不同类型的节点,如函数、变量等,它们之间可能有不同类型的依赖关系和数据流。通过将合约代码表示为异构图,可以更全面地描述合约的结构和行为,从而为漏洞检测提供更丰富的信息。

2 方法

本方法主体架构分为3个阶段:符号图^[22]的生成;异构图的构建;异构图神经网络模型的构建。

2.1 符号图的生成

Allamanis 等^[22]研究表明,程序可以转化为符号图表示,这种方法可以保留程序元素之间的语义关系。由于在漏洞检测中变量和函数是产生漏洞的基本因素,因此从源码中提取变量节点、函数节点和回退节

点,回退函数是智能合约中特有的,也是较多漏洞的根源。同时也提取数据流、控制流两种类型的边。

对提取的所有节点的符号化命名规则如下:首先将回退节点命名为 F。其次,根据变量节点和函数节点出现的顺序依次命名为“N-#”和“C-#”的形式,其中 #表示序号的数字。图 1 为由智能合约转化得到的符号图示例。

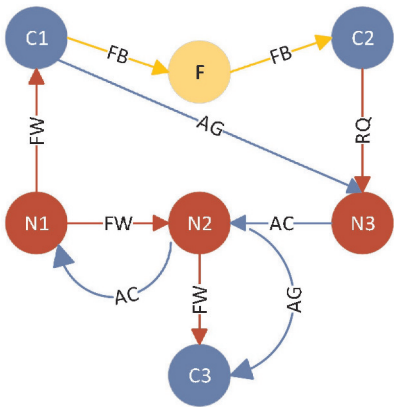


图 1 由智能合约转化得到的符号图

对提取的所有边也需要进行符号化命名,以方便后续边特征的提取,其命名方式如表 1 所示。

表 1 边符号化对应表

原始语句	符号化	边类型
assert	AS	控制流
require	RQ	
revert	RT	
throw	TH	
if	IF	
if else	IE	
if then	IT	
while do	WD	
for do	FD	
forward	FW	数据流
fallback	FB	
assign	AG	
access	AC	

2.2 异构图的构建

在异构图中,元路径是一种关键的方法,用于获取语义信息以及建模节点之间的关联。本文构造了 6 种元路径,每条元路径都有其独特的作用。

(1) Func-Control-Func:通过控制流连接两个函数节点。控制流在代码中扮演重要角色,揭示程序执行逻辑和控制结构。使用这条元路径,能够捕获函数之间的调用关系,分析代码中的条件语句、循环和分支等控制结构。

(2) Func-Data-Func:通过数据流连接两个函数节点。数据流关系描述了数据在函数之间的传递和使用方式。通过这条元路径,能够提取出函数之间的数据交互,识别数据的输入、输出和传递过程。

(3) Var-Control-Var:通过控制流连接两个变量节点。变量在程序中扮演着重要角色,通过控制流连接的变量节点可以更好提取出在条件语句、循环和分支等控制结构中变量值的变化过程。

(4) Var-Data-Var:通过数据流连接两个变量节点。数据流关系对于代码分析也十分关键,通过这条元路径,可以了解变量之间的数据传递和共享。

(5) Func-Control-Var:通过控制流连接函数和变量节点。结合函数和变量之间的控制流关系,可以理解函数执行过程中变量值的变化情况,以及函数执行的条件和分支,有助于进一步分析代码的行为和逻辑。

(6) Func-Data-Var:通过数据流连接函数和变量节点。结合函数和变量之间的数据依赖关系,可以了解函数执行过程中对变量的数据操作,推断代码的功能和数据处理过程,从而深入理解代码的语义。

在异构图中,除了元路径,边的特征和节点特征也是至关重要的。本文采用词袋模型作为基础方法来得到边和节点的初始向量表示,通过构造词汇映射字典将符号化后的点和边进行映射转化为对应的独热编码。

2.3 异构图神经网络模型

传统的图神经网络中,边和节点之间的关联性是相同的,无法反映复杂的程序结构信息。异构图中的元路径可以捕捉不同类型节点和边之间复杂的语义关系,例如函数之间的控制流和数据流连接,变量之间的控制流和数据流关联等。通过使用多条元路径,异构图能够更全面地表达复杂系统中多样化的信息和关联,使图数据的表示更加丰富多样。异构图允许不同类型节点和边采用不同的嵌入方法和特征表示,因此在处理不同类型的实体时可采用更适合的表示学习方法。这使异构图网络在处理多样化数据时具有更高的灵活性和适应性。

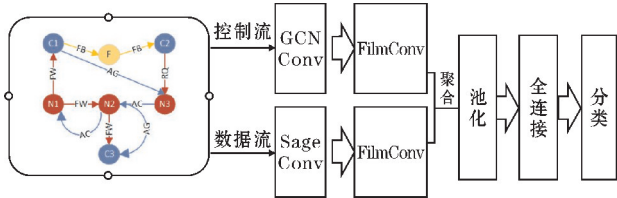


图 2 HG-SMD 模型结构

本文模型 HG-SMD(heterogeneous graph-smart contract detection)如图 2 所示,分别对不同类型的点边关系进行学习,将数据流的点边关系通过 Sage 层^[23]随机采样邻居节点的信息并进行聚合和整合学习节点在图中的表示:

$$\mathbf{X}'_i = \mathbf{W}_1 \mathbf{x}_i + \mathbf{W}_2 \cdot \text{mean}_{j \in N(i)} \mathbf{X}_j \quad (1)$$

其中, \mathbf{X}_i 表示节点 i 在当前层的表示特征; $N(i)$ 表示节点 i 的邻居节点的集合; \mathbf{X}_j 表示节点 j 的特征, 节点 j 是随机采样的一个邻居节点; \mathbf{W}_1 和 \mathbf{W}_2 分别是当前节点和邻居节点特征的权重矩阵。

由于在大多数情况中控制流的边较多, 因此将控制流的点边关系通过 GCNConv^[24] 将邻居节点信息进行聚合来学习图中的表示:

$$\mathbf{X}' = \hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}} \mathbf{X} \boldsymbol{\theta} \quad (2)$$

其中: $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ 表示的是插入自环的邻接矩阵; $\hat{\mathbf{D}}$ 表示每个节点的度矩阵, 度矩阵是一种用于描述图中节点连接情况的矩阵, 在 GCNConv 中度矩阵是一个对角矩阵, 其中对角线上的元素是节点的度 (邻居数量); \mathbf{X} 表示节点的特征矩阵; $\boldsymbol{\theta}$ 表示权重矩阵。相较于 Sage 层, GCN 卷积运算符在计算过程中使用对角线度矩阵对邻接矩阵进行归一化, 考虑了节点的度信息, 因此在度较大的情况下使用 GCNConv 来学习更具优势。

为使模型可以在每个特征图上学到更多的空间变化和语义信息, 并提高模型的泛化能力, 加入 Film 层^[25], 根据输入的上下文信息对每个图进行不同的平移和缩放, 从而增强模型对输入数据的表达能力。Film 层的计算为:

$$\mathbf{X}'_i = \sum_{r \in R} \sum_{j \in N(i)} \sigma(\gamma_{r,i} \odot \mathbf{W}_r \mathbf{X}_j + \beta_{r,i}) \quad (3)$$

其中: r 表示所在层数; \mathbf{W}_r 表示 r 层的权重矩阵; $\gamma_{r,i}$ 和 $\beta_{r,i}$ 表示缩放和平移的参数; σ 表示激活函数。从式(3)可以看到, 通过学习参数 $\gamma_{r,i}$ 模型可以自适应地调整每个节点对邻居节点特征的关心程度, 从而提高特征传播的表达能力。

通过对两种不同类型的边分别进行训练, 将得到的节点特征进行相加聚合, 可以得到每个节点最终的嵌入。这种方法能够综合考虑不同类型边的信息, 从而更全面地表示节点的特征和关系。

模型使用池化层将卷积处理后输出的节点特征集合, 进一步转换为一个表示整个图的全局特征。模型通过式(4)的方法将所有的节点进行聚合得到图级特征。

$$\mathbf{G} = \sum_{n=1}^N \mathbf{x}_n \quad (4)$$

最后, 采用全连接层进行分类任务, 将节点的综合特征映射为二元输出, 其中 0 表示节点不具有漏洞, 1 表示节点具有漏洞。

3 实验

3.1 实验数据

实验使用两种基准数据集对实验方法进行评估, 数据集为以太坊智能合约数据集 (ESC) 以及 VNT 链

智能合约数据集 (VSC)。

ESC^[19-20]: 该数据集包含约 40932 个以太坊智能合约, 大约有 307396 个函数。这些函数中, 大约有 5013 个函数至少具有一次调用 call. value 函数, 这些合约可能受到可重入漏洞影响。大约有 4833 个函数包含 block. timestamp 语句, 这些合约容易受到时间戳依赖漏洞影响。Qian 等^[20-21, 26] 通过专家规则及现有工具, 对可能受漏洞影响的合约进行标记, 并将该数据集划分成可重入漏洞和时间戳依赖漏洞两个子数据集。

VSC^[19-20]: 该数据集包含 4170 个 VNT 链智能合约, 有 13761 个函数, 其中 2975 个函数具有循环语句, 并且通过检测工具和专家规则对智能合约进行了标注。

3.2 实验设计

对数据集随机取 20% 的合约作为测试集, 其余为训练集。ESC 数据集用于对可重入漏洞、时间戳依赖漏洞检测; VSC 数据集用于对无限循环漏洞进行检测。

实验中, 采用准确率 (Acc), 召回率 (Recall), 精度 (Precision), F1 分数作为指标评估模型在测试集上的效果。

$$\text{Acc} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FP} + \text{FN} + \text{TN})} \quad (5)$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \quad (6)$$

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \quad (7)$$

$$\text{F1} = \frac{(2 \cdot \text{Precision} \cdot \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (8)$$

其中, TP 表示预测正确的正样本, FP 表示预测错误的正样本, FN 表示预测错误的负样本, TN 表示预测正确的负样本。

实验 1: 为验证本文方法的有效性, 对于 ESC 数据集, 选取 4 种常见以太坊智能合约漏洞检测工具 SmartCheck、Oyente、Securify 和 Slither 检测可重入漏洞和时间戳依赖漏洞, 并将结果与本文方法进行对比分析; 对于 VSC 数据集, 选取 4 种常见的无限循环检测工具 Jolt、PDA、SMT 和 Looper 对无限循环漏洞进行检测, 并将结果与本文方法进行对比分析。

实验 2: 为检验图神经网络在提取合约上下文以及语义信息的方式的效果, 选择两种用于漏洞识别的传统神经网络在两个数据集 ESC 和 VSC 上进行对比。这两个神经网络分别是循环神经网络 (RNN) 和长短期记忆网络 (LSTM)。它们在训练模型和检测漏洞前, 需要将合约转化为一维文本序列。

实验 3: 为验证本文模型基于异构图的方法在提取出不同节点和边的复杂语义信息方面的有效性, 选

取两种基于同构图的模型 TMP 和 AMP,分别在两个数据集上进行对比实验。

3.3 实验分析

3.3.1 HG-SMD 与现有漏洞检测工具的比较

实验结果如表 2、3 所示。通过结果可以看到,本文模型 HG-SMD 对可重入、时间戳依赖、无限循环 3 种漏洞进行检测的准确率分别为91.8%、87.04%、75.28%,工具 SmartCheck 对可重入、时间戳依赖两种漏洞的检测准确率为52.97%、44.32%。工具 Oyente 对可重入漏洞以及时间戳依赖的检测准确率为61.62%、59.45%。Securify 对于可重入漏洞的检测准确率为71.89%。Jolt 对于无限循环漏洞的检测准确率为42.88%,PDA 对无

限循环漏洞的检测准确率为46.44%,SMT 和 Looper 对无限循环漏洞检测的准确率分别为54.04%、59.56%。造成现有工具检测效率不佳的原因是由于传统的检测工具使用基于规则或静态分析的方法来提取特征,无法充分捕获合约代码的复杂性。而基于图神经网络的方法可以自动学习更丰富的特征表示,从而提高检测效果。智能合约通常包含复杂的逻辑和数据流,而且使用高级语言编写,例如 Solidity。这些合约的复杂性使传统方法难以全面理解和检测其中的漏洞;而基于图神经网络的方法可将合约表示为图形结构,并利用图神经网络的能力来处理复杂的数据流和控制流,因此能够更好地捕捉合约的特性。

表 2 HG-SMD 与常用工具在 ESC 上实验结果

单位:%

方法	可重入漏洞				时间戳依赖漏洞			
	准确率	召回率	精度	F1 分数	准确率	召回率	精度	F1 分数
SmartCheck	52.97	32.08	25.00	28.10	44.32	37.25	39.16	38.18
Oyente	61.62	54.71	38.16	44.96	59.45	38.44	45.16	41.53
Securify	71.89	56.60	50.85	53.57	—	—	—	—
Slither	77.12	74.28	68.42	71.23	74.20	72.38	67.25	69.72
HG-SMD	91.80	92.7	96.55	94.56	87.04	90.64	86.42	88.47

表 3 HG-SMD 与常用工具在 VSC 上实验结果

单位:%

方法	无限循环漏洞			
	准确率	召回率	精度	F1 分数
Jolt	42.88	23.11	38.23	28.81
PDA	46.44	21.73	42.96	28.26
SMT	54.04	39.23	55.69	45.98
Looper	59.56	47.21	62.72	53.83
HG-SMD	75.28	74.75	75.12	74.93

3.3.2 HG-SMD 与传统神经网络模型的比较

实验结果如表 4、5 所示。其中,RNN 对 3 种漏洞进行检测的准确率分别为49.64%、49.77%、49.57%,

LSTM 进行检测的准确率分别为 53.68%、50.79%、51.28%,其准确率远低于 HG-SMD 模型的91.80%、87.04%、75.28%。这是由于智能合约通常是由一系列复杂的控制流和数据流组成的,传统神经网络在处理这种非线性结构时表现较为有限。而图神经网络天生适合处理图形结构数据,能够更好地捕捉合约代码中的复杂依赖关系和交互模式。智能合约中的漏洞往往涉及多个代码的交互和依赖关系,传统神经网络在信息聚合方面的能力较为有限。相比之下,图神经网络采用消息传递和图卷积等方法,能够有效地聚合多个代码块的信息,从而更全面地分析合约漏洞。

表 4 HG-SMD 与传统神经网络在 ESC 上实验结果

单位:%

方法	可重入漏洞				时间戳依赖漏洞			
	准确率	召回率	精度	F1 分数	准确率	召回率	精度	F1 分数
RNN	49.64	58.78	49.82	50.71	49.77	44.59	51.91	45.62
LSTM	53.68	67.82	51.65	58.64	50.79	59.23	50.32	54.41
HG-SMD	91.80	92.7	96.55	94.56	87.04	90.64	86.42	88.47

表 5 HG-SMD 与传统神经网络在 VSC 上实验结果

单位:%

方法	无限循环漏洞			
	准确率	召回率	精度	F1 分数
RNN	49.57	47.86	42.10	44.79
LSTM	51.28	57.26	44.07	49.80
HG-SMD	75.28	74.75	75.12	74.93

3.3.3 HG-SMD 与同构图神经网络模型的比较

实验结果如表 6、7 所示。TMP 模型在 3 种漏洞的检测上准确率分别达到了84.48%、83.45%、74.61%,AMP 在 3 种漏洞的检测上准确率分别达到了 90.19%、86.52%、80.32%。在 ESC 数据集上对可重入漏洞、时间戳依赖漏洞的检测本文模型是要优于

TMP 和 AMP 的相较于 TMP,HG-SMD 准确率分别升了 7.32 个百分点和3.59个百分点,相较于 AMP,HG-SMD 准确率分别升了1.61个百分点和0.52个百分点。在 VSC 数据集上对无限循环漏洞的检测较 TMP 提高了 0.67 个百分点。造成该结果是因为智能合约通常包含不同类型的代码块(如函数、变量、条件、循环等)。基于异构图的模型允许对不同类型的节点和边进行建模,能够更好地捕捉不同代码块之间的关联关系和特

征。相比之下,普通图神经网络可能对所有节点和边都采用相同的特征表示,可能无法充分利用不同代码块的特异性。而 HG-SMD 在无限循环漏洞的检测上低于 AMP 可能是由于专家规则对于无限循环漏洞非常敏感,能够有效地指出引起该漏洞的重要函数和变量,从而指导模型关注合约代码的重要部分,以提高漏洞检测的准确性。

表6 HG-SMD 与 AMP、TMP 在 ESC 上实验结果

单位:%

方法	可重入漏洞				时间戳依赖漏洞			
	准确率	召回率	精度	F1 分数	准确率	召回率	精度	F1 分数
TMP	84.48	82.63	74.06	78.11	83.45	83.82	75.05	79.19
AMP	90.19	89.69	86.25	87.94	86.52	86.23	82.07	84.10
HG-SMD	91.80	92.70	96.55	94.56	87.04	90.64	86.42	88.47

表7 HG-SMD 与 AMP、TMP 在 VSC 上实验结果

单位:%

方法	无限循环漏洞			
	准确率	召回率	精度	F1 分数
TMP	74.61	74.32	73.89	74.10
AMP	80.32	79.08	78.69	78.88
HG-SMD	75.28	74.75	75.12	74.93

4 结束语

提出一种基于异构图的智能合约漏洞检测方法。该方法通过将智能合约源码转化为具有多种节点以及两种边类型的异构图,通过图神经网络的训练方式将不同的点边关系经过卷积层进行学习,最后得到图级的特征表示,再通过全连接层进行分类。本文提出的模型 HG-SMD 在 ESC 和 VSC 两种数据集上进行测试,在多种类型的漏洞检测上都要优于现有的工具,充分展现了其有效性。

未来,将进一步扩展漏洞数据集,使覆盖更多种类型的漏洞,以验证本文方法的泛化性。此外,还计划探索节点级的预测方法,以便更精确地定位漏洞所在。同时,还可以通过引入注意力机制来提升模型的可解释性,以便更好地理解模型对漏洞的判别依据。

参考文献:

[1] Huang K,Mu Y,Rezaeibagha F,et al. Design and analysis of cryptographic algorithms in blockchain [M]. Crc Press,2021.

[2] Nakamoto S.Bitcoin: A peer-to-peer electronic cash system[C]. Decentralized business review,2008.

[3] Li X,Jiang P,Chen T,et al. A survey on the secu-

rity of blockchain systems[J]. Future generation computer systems,2020,107: 841–853.

[4] Chen T,Li X,Wang Y,et al. An adaptive gas cost mechanism for ethereum to defend against under-priced dos attacks[C]. Information Security Practice and Experience: 13th International Conference, ISPEC 2017, Melbourne, VIC, Australia, December 13–15,2017,Proceedings 13. Springer International Publishing,2017: 3–24.

[5] 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,2016:254–269.

[6] Kalra S,Goel S,Dhawan M,et al. Zeus: analyzing safety of smart contracts[C]. Ndss,2018: 1–12.

[7] Bhargavan K,Delignat-Lavaud A,Fournet C,et al. Formal verification of smart contracts: Short paper [C]. Proceedings of the 2016 ACM workshop on programming languages and analysis for security, 2016: 91–96.

[8] Yang Z,Lei H. Fether: An extensible definitional interpreter for smart-contract verifications in coq [J]. iee access,2019,7: 37770–37791.

[9] 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, 2018: 259–269.

[10] Chen T,Zhang Y,Li Z,et al. Tokenscope: Automatically detecting inconsistent behaviors of cryptocurrency tokens in ethereum[C]. Proceedings of the 2019 ACM SIGSAC conference on computer

- and communications security,2019:1503–1520.
- [11] 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,2018:67–82.
- [12] Tikhomirov S, Voskresenskaya E, Ivanitskiy I, et al. Smartcheck: Static analysis of ethereum smart contracts[C]. Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain,2018: 9–16.
- [13] Feist J, Grieco G, Groce A. Slither: a static analysis framework for smart contracts [C]. 2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). IEEE,2019: 8–15.
- [14] Carbin M, Misailovic S, Kling M, et al. Detecting and escaping infinite loops with jolt[C]. ECOOP 2011 – Object-Oriented Programming: 25th European Conference, Lancaster, Uk, July 25 – 29, 2011 Proceedings 25. Springer Berlin Heidelberg,2011: 609–633.
- [15] Ibing A, Mai A. A fixed-point algorithm for automated static detection of infinite loops[C]. 2015 IEEE 16th International Symposium on High Assurance Systems Engineering. IEEE,2015:44–51.
- [16] Kling M, Misailovic S, Carbin M, et al. Bolt: on-demand infinite loop escape in unmodified binaries[J]. ACM SIGPLAN Notices,2012,47(10): 431–450.
- [17] Burnim J, Jalbert N, Stergiou C, et al. Looper: Lightweight detection of infinite loops at runtime [C]. 2009 IEEE/ACM International Conference on Automated Software Engineering. IEEE, 2009: 161–169.
- [18] Zhuang Y, Liu Z, Qian P, et al. Smart contract vulnerability detection using graph neural networks [C]. Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence,2021:3283–3290.
- [19] Liu Z, Qian P, Wang X, et al. Smart contract vulnerability detection: from pure neural network to interpretable graph feature and expert pattern fusion[J]. arXiv preprint arXiv:2106.09282:2021.
- [20] Zhang C, Song D, Huang C, et al. Heterogeneous graph neural network[C]. Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining,2019:793–803.
- [21] Allamanis M, Brockschmidt M, Khademi M. Learning to represent programs with graphs[J]. arXiv preprint arXiv:1711.00740,2017.
- [22] Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs [J]. Advances in neural information processing systems, 2017,30.
- [23] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[J]. arXiv preprint arXiv:1609.02907,2016.
- [24] Brockschmidt M. Gnn-film: Graph neural networks with feature-wise linear modulation[C]. International Conference on Machine Learning. PMLR,2020: 1144–1152.
- [25] Qian P, Liu Z, He Q, et al. Towards automated reentrancy detection for smart contracts based on sequential models [J]. IEEE Access, 2020, 8: 19685–19695.

Smart Contract Vulnerability Detection based on Heterogeneous Graph

HOU Yishan, WANG Yi

(College of Cybersecurity, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: To address that the existing smart contract vulnerability detection-based deep learning cannot effectively use context information, This paper proposes a smart contract vulnerability detection based on a heterogeneous graph, Which parses the contract into a Symbol diagram containing data-flow edge and control-flow edge. Then it uses graph neural networks to perform representation learning on the graph, finally, the vulnerability prediction is performed through the neural networks. Experiments are conducted on ESC and VSC data sets, and comparing them with existing tools and models, the results show that the method has improved in the four indicators of accuracy, recall, precision, and F1-score.

Keywords: smart contract; vulnerability detection; deep learning; graph neural network