基于预训练和混合神经网络的智能合约漏洞检测

张光华,张思怡,高业萍,武少广
(河北科技大学 信息科学与工程学院,河北 石家庄 050018)

摘　要:针对现有智能合约漏洞检测方法准确率低以及对合约源代码特征提取不足的问题,该文提出一种基于预训练 BERT 模型(Bidirectional Encoder Representations from Transformer,BERT)和混合神经网络串行的智能合约漏洞检测方案 SCVD-PBHNN。首先,对源代码进行数据预处理,去除冗余信息,保留源代码中的关键语句信息;其次,利用 BERT 模型替换了传统的静态词嵌入模型,深入挖掘智能合约源代码的语义特征;然后,结合文本卷积神经网络(Text Convolutional Neural Network,TextCNN)和双向长短期记忆网络(Bi-directional Long Short-Term Memory,BiLSTM)混合神经网络构建特征提取层,对传入的词向量进行训练,充分提取源代码上下文信息和局部关键特征;最后,通过激活函数对特征向量进行归一化处理,实现漏洞检测与识别。实验结果表明,该方案在准确率、精确率、召回率以及 F1 值等评价指标上相比已有方案均有明显提升,能准确识别四种常见的漏洞,准确率分别为 94.40%、93.72%、96.29%、93.53%。

关键词:混合神经网络;BERT;TextCNN-BiLSTM;智能合约;漏洞检测
中图分类号:TP309　　**文献标识码**:A　　**文章编号**:1673-629X(2025)01-0094-07
doi:10.20165/j.cnki.ISSN1673-629X.2024.0299

Smart Contract Vulnerability Detection Based on Pre-training and Hybrid Neural Networks

ZHANG Guang-hua,ZHANG Si-yi,GAO Ye-ping,WU Shao-guang
(School of Information Science and Engineering,Hebei University of Science Technology,
Shijiazhuang 050018,China)

Abstract:Aiming at the problems of low accuracy of existing smart contract vulnerability detection methods and insufficient feature extraction of smart contract source code,we propose a new method based on pre-trained BERT model (Bidirectional Encoder Representations from Transformer, BERT) and hybrid neural network serial smart contract vulnerability detection scheme SCVD-PBHNN. Firstly,the source code is preprocessed to remove redundant information and retain key statement information. Secondly,BERT model is used to replace the traditional static word embedding model and dig deep into the semantic features of smart contract source code. Then,combined with Text Convolutional Neural Network (TextCNN) and Bi-directional Long Short-Term Memory (BiLSTM) hybrid neural network to construct feature extraction layer,the incoming word vectors are trained to fully extract source code context information and local key features. Finally,the feature vector is normalized by activation function to realize vulnerability detection and identification. The experimental results show that compared with the existing schemes,the proposed scheme has significantly improved in the evaluation indexes of accuracy,precision,recall rate and F1 value,which can accurately identify four common vulnerabilities,with accuracy rates of 94.40%,93.72%,96.29% and 93.53%,respectively.

Key words:hybrid neural networks;BERT;TextCNN-BiLSTM;smart contract;vulnerability detection

0　引　言

智能合约最早由美国学者尼克·萨博(Nick Szabo)提出,是一种计算机协议,其目的是通过加密协议和数字安全机制来传递、确认或实施合同条款[1]。然而,由于缺乏可信的执行环境,当时的探索仅停留在理论层面,未能引起广泛关注。随着智能合约与区块链技术的融合,才被广泛使用。以太坊作为区块链的核心平台,提供了稳定的智能合约执行环境,许多去中心化的应用都是通过在以太坊上部署智能合约的方式来实现运行的[2]。

收稿日期:2024-06-16　　**修回日期**:2024-10-17
基金项目:国家自然科学基金(62072239,62372236)
作者简介:张光华(1979-),男,博士,教授,CCF 高级会员(51334S),研究方向为网络与信息安全;通讯作者:武少广(1987-),男,硕士,讲师,研究方向为深度学习在网络安全领域的研究与应用。

由于智能合约的编写涉及复杂的逻辑和重要的功能,设计不良的智能合约会暴露漏洞,攻击者可利用这些漏洞谋取利益,导致其在安全性方面存在难以保障的风险[3]。2017 年,Parity 多重签名钱包由于一段库合约代码被错误地删除,导致大量资金被永久冻结[4]。2018 年,美链遭受整数溢出攻击,导致代币供应量无限膨胀,价格缩水至零。在经历众多的安全事件后,越来越多的安全漏洞被非法用户利用,这些安全问题严重威胁了区块链的发展,并使智能合约陷入了信任危机,因此,在部署智能合约之前,必须对其进行全面的安全漏洞检测和代码审核,以确保其安全性[5]。

现阶段,研究人员提出多种智能合约漏洞检测方法,包括符号执行、模糊测试、形式化验证、机器学习等[6]。这些传统的漏洞检测方法无法从合同中自动提取有用的特征,往往依赖专家设定的规则,检测漏洞效率低下。为了高效地检测智能合约中包含的漏洞,该文提出基于 BERT 和混合神经网络串行的漏洞检测方案 SCVD-PBHNN,该方案能有效地提取智能合约源代码特征,为神经网络模型提供更丰富的语义特征。与传统漏洞检测方法相比,该检测方案能检测到四种类型的漏洞,检测准确率高。

1　相关工作

现有的智能合约漏洞检测方法可以分为两类:基于传统技术的漏洞检测方法和基于深度学习技术的漏洞检测方法。传统的漏洞检测方法主要分为符号执行、模糊测试、形式化验证、程序分析和污点分析等。符号执行[7]是将程序中的变量和输入值表示为符号,并通过逐条解释执行程序指令的过程中更新执行状态。Oyente[8]是利用符号执行进行漏洞检测的代表性工具,可检测多种漏洞。模糊测试通过构造随机测试用例,监视程序执行过程中的异常行为来判断程序是否存在安全隐患[9]。Jiang 等提出模糊测试框架 ContractFuzzer[10],用于自动检测智能合约中的潜在漏洞。Nguyen 等[11]提出基于反馈指导的适应性模糊测试技术 sFuzz。形式化验证是使用严格的数学逻辑和形式规范来验证程序的属性和安全性。E. Hildenbrandt 等[12]提出了形式语义框架 KEVM,其提供了完整 EVM 语义的可执行和可读的形式模型。程序分析通过自动分析程序的特征和属性来确定其安全性,代表性的智能合约漏洞检测工具和框架包括 SmartCheck[13],Slither[14]和 Vanda[15]。污点分析是在程序执行过程中通过标记源代码或字节码中的关键值,分析其数据流来判断是否存在漏洞。上述传统的漏洞检测方法受限于预先定义的规则,漏洞检测准确率低。

基于深度学习的漏洞检测方法能够解决传统工具的不足,大量的研究人员使用深度学习的方法来检测智能合约的漏洞[16]。Zhuang 等人使用图神经网络来检测智能合约漏洞[17],该方法针对漏洞分别实现合约图的生成,但移植和扩展困难,检测精度有限。Wang 等人提出基于机器学习的漏洞检测框架 ContractWard[18],利用支持向量机、神经网络、随机森林和 K 近邻算法分析智能合同漏洞。Yu 等人[19]提出一种漏洞检测框架 DeeSCVHunter,该方案提出漏洞候选切片的概念,并利用三个单词嵌入模型 Word2vec、FastText 和 GloVe 训练处理好的切片,可以有效地检测合约漏洞。孟祥爱[20]提出基于 BERT 模型和双向长短期记忆网络的方法对智能合约操作码进行漏洞检测。Jeon 等人[21]使用 BERT 模型进行合同漏洞检测,并提出了一种 SmartConDetect 的方案来检测漏洞。李鑫等[22]提出基于序列特征和图特征的双通道并行处理的漏洞检测方法检测智能合约漏洞。林彦君等[23]提出基于语义词嵌入和图结构特征相融合的漏洞检测方法,能有效地提取智能合约的特征,但该方法只检测整数溢出漏洞,检测类型单一。结果表明,使用深度学习进行漏洞检测优于传统漏洞检测方法,但仍有很大的改进空间。为了解决上述智能合约漏洞检测方法的不足,该文提出一种新的检测方案 SCVD-PBHNN。该方案首先使用 BERT 模型提取词向量,然后构建由 TextCNN 和 BiLSTM 组成的混合神经网络作为特征提取层。通过捕获源代码中不同句子的语义信息,提高漏洞检测的准确率,识别多种类型的漏洞。

2　漏洞检测模型

该文提出的 SCVD-PBHNN 方案的整体框架如图 1 所示,主要分为四个部分。首先对智能合约源代码进行预处理,去除无用信息,并将其转化为代码序列;其次采用 BERT 模型将预处理后的智能合约源代码序列编码为词嵌入矩阵;然后将输出的词向量输入到特征提取层进行训练,捕获源代码序列中的特征;最后漏洞检测层输出检测结果。

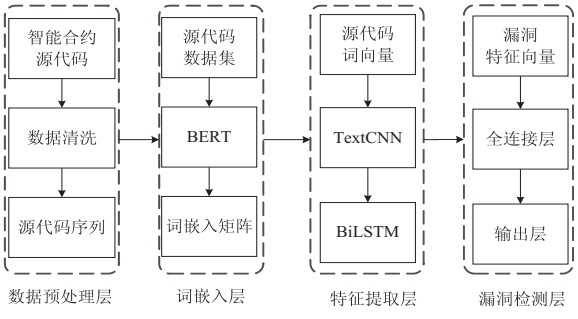

图 1　漏洞检测整体框架

2.1　数据预处理层

该文使用的数据集是以太坊平台上真实的智能合

约源代码。在处理智能合约源代码过程中,要考虑源代码的特点。然而在大多数情况下,导致漏洞的代码行总是集中在少数几行,且大多数的智能合约源代码中存在大量与漏洞无关的信息。这些冗余信息为智能合约的矢量化表示引入了噪声,从而对分类效果产生影响。为了减少噪声对检测结果的干扰,对合约源代码进行清理。处理过程如图 2 所示。

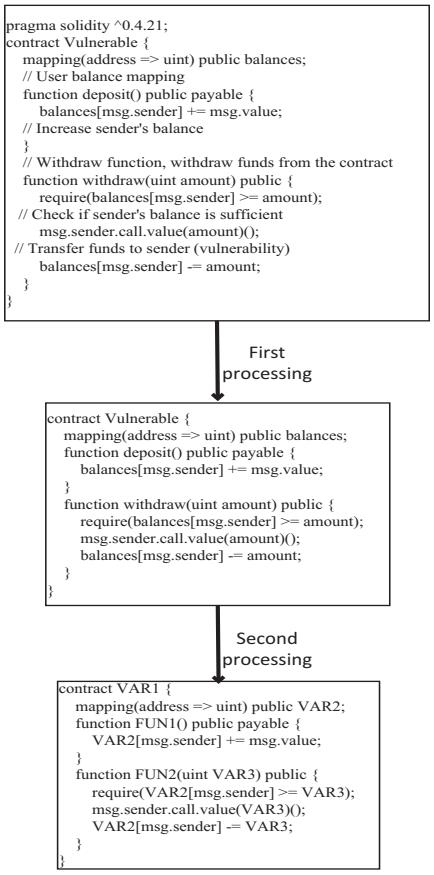

图 2 源代码预处理过程

在步骤 1 中,首先删除 Solidity 代码版本号,代码版本号通常位于合约的第一行,与漏洞无关;其次删除合约中的注释、制表符、非 ASCII 值和空白行,这些字符增加了向量化表示的维度,删除这些无关信息不会影响结果。为了规范化代码片段并避免不同命名规则的潜在影响,在步骤 2 中,将每个代码片段中用户自定义的变量名表示为 VAR 加数字(例如“VAR1”、“VAR2”),用户自定义函数名表示为 FUN 加数字(例如“FUN1”、“FUN2”),然后删除智能合约中所有空格,这样可以保留源代码关键语义。

2.2 词嵌入层

智能合约源代码中每个词都可能包含上下文语义关系,词向量应该根据上下文语义来表示,预先训练好的 BERT 语言模型可以动态地嵌入源代码的单词表示,并且可以很大程度上解决信息丢失的问题。BERT 模型是基于双向 Transformer 编码器构建的,由自注意力机制和前馈神经网络组成。BERT 结构如图 3 所示。

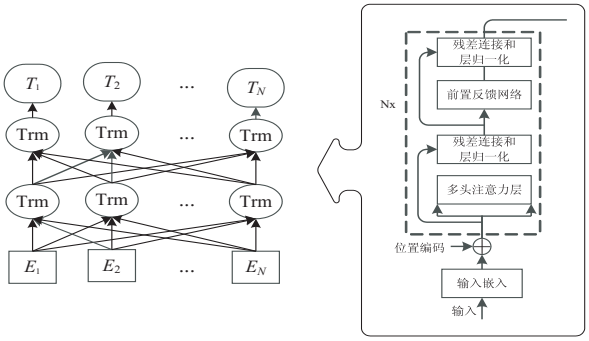

图 3 BERT 结构

为了更好地捕获单词的顺序信息和上下文关系,引入位置编码表示单词所在位置,如公式 1 和公式 2 所示。

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin\left(\frac{\mathrm{pos}}{10\,000^{\frac{2i}{d}}}\right) \tag{1}$$

$$\mathrm{PE}_{(\mathrm{pos},2i+1)} = \cos\left(\frac{\mathrm{pos}}{10\,000^{\frac{2i}{d}}}\right) \tag{2}$$

其中,pos 是单词的位置,i 是单词向量的维度位置,d 是输出维度。Transformer 模型通常使用多个并行的注意力头来处理输入序列。每个注意力头都是一个独立的自注意力层。对于每个输入的词向量 $\boldsymbol{x}_i$,通过相关联的权重矩阵 $\boldsymbol{W}_Q$、$\boldsymbol{W}_K$、$\boldsymbol{W}_V$ 来计算得到其对应的查询向量 $\boldsymbol{Q}_i$、键向量 $\boldsymbol{K}_i$ 和值向量 $\boldsymbol{V}_i$,如公式 3 ~ 5 所示。

$$\boldsymbol{Q}_i = \boldsymbol{W}_Q * \boldsymbol{x}_i \tag{3}$$

$$\boldsymbol{K}_i = \boldsymbol{W}_K * \boldsymbol{x}_i \tag{4}$$

$$\boldsymbol{V}_i = \boldsymbol{W}_V * \boldsymbol{x}_i \tag{5}$$

使用 softmax 归一化处理得到注意力权重 a_i,将注意力权重乘值向量 $\boldsymbol{V}_i$,得到自注意力头的输出向量 $\mathbf{head}_i$。如公式 6 和公式 7 所示。

$$a_i = \mathrm{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{k}_i^{\mathrm{T}}}{\sqrt{D_k}}\right) \tag{6}$$

$$\mathbf{head}_i = a_i * v_i \tag{7}$$

将输入矩阵 $\boldsymbol{X}$ 分别传递到不同自注意力层,多头注意力将注意力头生成的向量叠加和拼接起来,产生包含所有注意力头部信息的向量矩阵;接着将拼接后的矩阵乘权重矩阵 $\boldsymbol{W}^o$ 得到多头注意力的输出结果,如公式 8 所示。将输出结果经残差相加、归一化处理和前馈神经网络,最终获得词向量表示。

$$\mathrm{MultiHead} = \mathrm{concat}(\mathbf{head}_1, \cdots, \mathbf{head}_8)\boldsymbol{W}^O \tag{8}$$

2.3 特征提取层

该文提出了一种结合 TextCNN 和 BiLSTM 混合神经网络串行的深度学习模型用于提取源代码的特征。TextCNN 结构简单,主要由输入层、卷积层、池化层、全连接层这四部分组成。该文采用三种不同尺寸

的卷积核来提取源代码序列的局部特征,设置这些卷积核的大小分别为 2、3 和 4,首先将词向量输入卷积层,通过卷积操作得到特征信息;然后对每个卷积核生成的特征图经过最大池化操作,提取每个卷积核在输入序列上的最大值;最后,将每个卷积核提取到的最大值拼接在一起作为 BiLSTM 模型的输入。BiLSTM 是循环神经网络(RNN)的变体。LSTM 通过引入输入门、遗忘门、输出门以及记忆单元,有效解决了 RNN 中梯度消失和梯度爆炸等问题[24]。遗忘门 f_t、输入门 i_t、输出门 O_t、候选记忆单元 $\tilde{C}_t$ 如公式 9 ~ 12 所示。

$$f_t = \sigma(\boldsymbol{W}_{xf} \bullet x_t + \boldsymbol{W}_{hf} \bullet h_{t-1} + \boldsymbol{b}_f) \tag{9}$$

$$i_t = \sigma(\boldsymbol{W}_{xi} \bullet x_t + \boldsymbol{W}_{hi} \bullet h_{t-1} + \boldsymbol{b}_i) \tag{10}$$

$$O_t = \sigma(\boldsymbol{W}_{xo} \bullet x_t + \boldsymbol{W}_{ho} \bullet h_{t-1} + \boldsymbol{b}_o) \tag{11}$$

$$\tilde{C}_t = \tanh(\boldsymbol{W}_{xc} \bullet x_t + \boldsymbol{W}_{hc} \bullet h_{t-1} + \boldsymbol{b}_c) \tag{12}$$

式中,$\boldsymbol{W}$ 表示权重矩阵,b 表示偏移向量,σ 表示 Sigmoid 函数,h_{t-1} 表示 $t-1$ 时刻的输出,x_t 表示 t 时刻的输入。新的细胞状态 C_t 是利用遗忘门来过滤上一时刻细胞状态的信息,利用输入门来整合当前输入的信息得到的,如公式 13 所示。

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{13}$$

最终的隐藏状态 h_t 由当前细胞状态与输出门共同决定,如公式 14 所示:

$$h_t = O_t * \tanh(C_t) \tag{14}$$

BiLSTM 模型由两个独立的 LSTM 模型组成,一个用于处理正向序列信息,另一个捕捉反向序列信息[25]。相较于单向 LSTM 模型,BiLSTM 能更有效地捕捉上下文信息,提高分类准确率。因此,该文采用 BiLSTM。

2.4　漏洞检测层

漏洞检测层由全连接层和输出层两部分组成。全连接层采用 sigmoid 激活函数进行归一化处理,并施加适当的权重参数和偏移参数,激活函数如公式 15 所示。

$$y = \text{sigmoid}(w_i y_i + b) \tag{15}$$

其中,y_i 为特征提取层的输出,w_i 为权重参数,b 表示偏移。输出层将最终的漏洞检测结果以数字标签的形式呈现。

3　实验与分析

3.1　实验环境

实验运行环境为 Windows 10,CPU 为 Intel(R) Core(TM) i5-8300H,GPU 为 NVIDIA GeForce GTX 1050 Ti。

3.2　数据集

近年来,以太坊社区开发了大量工具来分析易受攻击的智能合约,但还没有标准的开源漏洞数据集。为了收集足够的实验数据,使用 SmartBug Wild[26] 和 CBGRU[27] 的数据集作为文中数据集。该文对数据集中包含的合约源代码进行审查,筛选掉一些编译器版本较低的智能合约。数据集最终被分为两类:有漏洞的智能合约和没有漏洞的智能合约。易受攻击的智能合约包含四种类型的漏洞,分别是堆栈调用深度攻击漏洞、整数下溢漏洞、可重入漏洞和无限循环漏洞。

3.3　评价指标

采用 4 个经典的指标进行评估,即准确率(Accuracy)、精确率(Precision)、召回率(Recall)以及 F1 值(F1)。准确率表示所有正确预测的样本数在总样本中所占的比例;精确率表示正类别的正确预测数占所有预测为正类别的样本数的比例;召回率表示正类别的正确预测数占所有实际正类别的样本数的比例;F1 值是精确度和召回率的调和平均数。计算如公式 16 ~ 19 所示。

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{16}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{17}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{18}$$

$$\text{F1} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \tag{19}$$

其中,真阳性(TP)表示将正类别样本预测为正类别;假阳性(FP)表示将负类别样本预测为正类别;真阴性(TN)表示将负类别样本预测为负类别;假阴性(FN)表示将正类别样本预测为负类别。

3.4　实验结果分析

为验证模型的有效性,将数据集按照 8∶2 的比例划分为训练集和测试集,从模型参数和模型性能来分析文中方案的实验结果。

3.4.1　模型参数设置

批量大小选择对模型的训练效果和泛化性能有显著影响。为了选择合适的批量大小,设置了 16、32、64、128 四种不同的批量大小参数进行实验,实验结果如图 4 所示。由图 4(a) ~ (d)可以看出,批量大小的不同对评价指标产生了明显影响。四种漏洞的准确率、精确率、召回率和 F1 值均随批量大小的增加而下降。通过分析结果发现,当批量大小设置为 16 时,各项指标表现最佳;而设置为 128 时,各项指标最低,模型训练效果最差。因此,该文最终选择将批量大小设置为 16。

3.4.2　模型性能分析

该文采用 Adam 优化器进行模型训练,设置训练轮数为 30 轮,批量大小为 16,学习率为 0.000 1,同时

将 dropout 设置为 0.5 防止过拟合。图 5 分别展示了四种漏洞准确率和损失值随着轮次的变化趋势。在训练过程中,模型的准确率随着轮次的增加一直呈上升趋势并最终趋于稳定,同时损失逐渐减小并保持稳定。

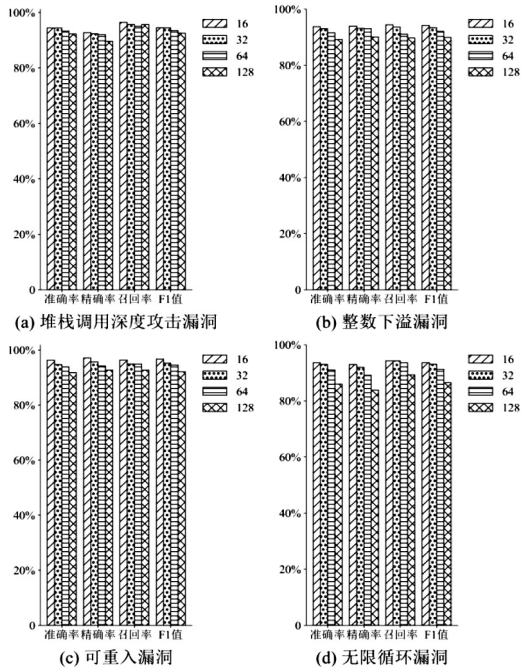

图 4 不同批量大小的实验结果

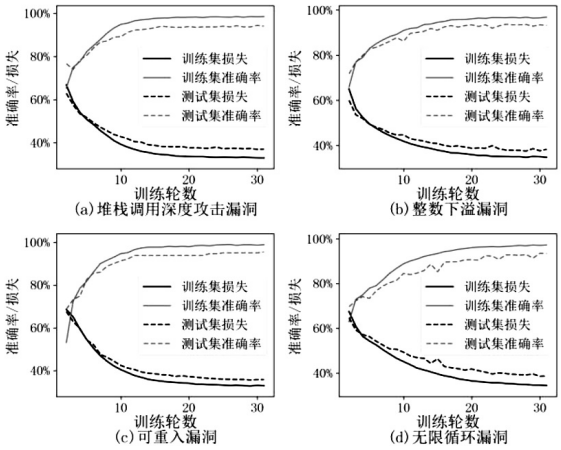

图 5 漏洞的准确率和损失值

表 1 为四种漏洞的检测结果。由表 1 可知,文中模型能检测出四种合约漏洞,四种漏洞的各项指标均较好,特别是可重入漏洞的精确率高达 97.05%。该文提出的方案充分提取源代码的特征,在检测四种智能合约漏洞方面表现优异,检测效果好。

表 1 不同漏洞评估结果 %

漏洞类型	准确率	精确率	召回率	F1 值
堆栈调用深度攻击漏洞	94.40	92.73	96.40	94.53
整数下溢	93.72	93.92	94.36	94.14
可重入漏洞	96.29	97.05	96.35	96.70
无限循环	93.53	92.93	94.26	93.59

3.5 消融实验

为验证文中方案中 TextCNN-BiLSTM 对模型性能的影响,进行消融实验。结果如表 2 所示,文中方案在各项指标上表现更优。

表 2 消融实验 %

漏洞类型	评价指标	模型		
		BERT-BiLSTM	BERT-TextCNN	SCVD-PBHNN
堆栈调用深度攻击	准确率	88.08	91.41	94.40
	精确率	90.45	90.97	92.73
	召回率	85.25	92.24	96.40
	F1 值	87.77	91.60	94.53
整数下溢漏洞	准确率	84.19	89.33	93.72
	精确率	84.72	89.37	93.92
	召回率	85.91	90.84	94.36
	F1 值	85.31	90.10	94.14
可重入漏洞	准确率	88.06	90.53	96.29
	精确率	88.02	93.18	97.05
	召回率	91.24	89.78	96.35
	F1 值	89.60	91.44	96.70
无限循环漏洞	准确率	77.73	87.79	93.53
	精确率	76.09	86.75	92.93
	召回率	81.00	89.24	94.26
	F1 值	78.47	87.98	93.59

总体上看 BERT-TextCNN 的检测性优于 BERT-BiLSTM。尤其在在检测可重入漏洞方面,文中模型的召回率比 BERT-BiLSTM 高了 5.11 百分点,且文中模型比 BERT-TextCNN 的准确率、精确率、F1 值分别提高了 5.76 百分点、3.87 百分点、5.26 百分点;在检测无限循环漏洞方面,文中模型的准确率、精确率、召回率和 F1 值比 BERT-TextCNN 分别提高了 5.74 百分点、6.18 百分点、5.02 百分点、5.61 百分点。这表明,单独使用 TextCNN 或 BiLSTM 模型后,模型性能均有所下降,而文中模型能综合两个模型的优势,提升了模型的整体性能。

3.6 对比实验

为验证文中模型漏洞检测的性能,将文中模型与其他方法进行对比,设置了两组实验。

第一组对比实验是用 BERT 模型和传统的静态词嵌入模型 Word2Vec、FastText 进行对比,结果如表 3 所示。静态词嵌入模型 Word2Vec、FastText 检测漏洞的四个指标均未达到 90%,仅 FastText 对可重入漏洞的精确率达到 91.01%。而文中模型相比于传统的词嵌入模型各项指标有大幅度提升。通过分析对比结果,证明 BERT 模型对合约源代码的特征表达能力优于静态词嵌入模型,弥补传统的词嵌入模型在源代码

表 3　对比其它词嵌入模型　%

漏洞类型	评价指标	模型		
		Word2Vec	FastText	SCVD-PBHNN
堆栈调用深度攻击	准确率	88.32	87.45	94.40
	精确率	89.35	88.54	92.73
	召回率	86.99	86.21	96.40
	F1 值	86.11	87.32	94.53
整数下溢漏洞	准确率	83.98	84.06	93.72
	精确率	82.54	82.27	93.92
	召回率	86.42	86.96	94.36
	F1 值	84.36	84.53	94.14
可重入漏洞	准确率	87.61	87.75	96.29
	精确率	89.77	91.01	97.05
	召回率	85.04	83.80	96.35
	F1 值	87.24	87.17	96.70
无限循环漏洞	准确率	81.93	82.12	93.53
	精确率	80.18	80.01	92.93
	召回率	84.94	85.73	94.26
	F1 值	82.44	82.74	93.59

特征提取方面的不足,能更好捕捉源代码中丰富的语义信息和上下文关系,检测漏洞准确率更高。

第二组是与深度学习模型进行对比。将文中模型与 RNN、LSTM、门控循环单元(GRU)、CNN-BiLSTM-Attention[20]、卷积神经网络(CNN)以及 BiLSTM-Attention[9]六种深度学习模型进行对比,结果如表 4 所示。文中模型的检测效果优于其它五种模型。从整体上看,CNN 在四个单独的模型中检测效果最好;相较于单独模型,CNN-BiLSTM-Attention 检测漏洞效果优于 BiLSTM-Attention。而文中模型检测漏洞的整体性能明显优于 CNN-BiLSTM-Attention,尤其是在可重入漏洞的检测中,准确率提高了 6.17 百分点。该文采用 TextCNN 和 BiLSTM 模型在模型架构上更为简洁、有效,在特征提取和顺序依赖建模之间达到平衡,避免模型过拟合或训练不足的问题,能更好捕获源代码的特征,因此该方案检测准确率高。

表 4　文中模型与深度学习方案的对比结果　%

漏洞类型	评价指标	模型						
		RNN	LSTM	GRU	CNN	BiLSTM-ATT	CNN-BiLSTM-ATT	SCVD-PBHNN
堆栈调用深度攻击	准确率	83.75	85.37	86.28	86.82	87.43	91.87	94.40
	精确率	79.19	82.29	89.14	83.82	89.53	90.87	92.73
	召回率	91.72	90.28	82.73	91.36	84.10	93.16	96.40
	F1 值	85.00	86.10	85.82	87.43	86.73	92.00	94.53
整数下溢漏洞	准确率	79.54	85.06	79.17	89.83	83.06	92.22	93.72
	精确率	79.55	84.80	78.26	92.38	85.92	92.82	93.92
	召回率	83.09	87.79	84.50	88.26	81.69	90.37	94.36
	F1 值	81.28	86.27	81.26	90.27	83.75	91.57	94.14
可重入漏洞	准确率	81.89	86.41	79.42	90.04	88.95	90.12	96.29
	精确率	82.51	85.61	80.41	94.57	88.10	94.48	97.05
	召回率	86.13	91.24	83.94	89.05	91.78	87.59	96.35
	F1 值	84.28	88.33	82.14	91.72	89.90	90.90	96.70
无限循环漏洞	准确率	78.09	78.09	77.73	88.86	78.94	89.76	93.53
	精确率	76.61	77.54	78.81	87.80	79.00	87.75	92.93
	召回率	81.00	79.21	75.98	90.32	79.13	92.47	94.26
	F1 值	78.74	78.36	77.37	89.04	79.06	90.05	93.59

4　结束语

智能合约漏洞检测在区块链安全中至关重要。为提高传统方法的性能,提出了一种基于预训练 BERT 模型和混合神经网络串行的智能合约漏洞检测方案 SCVD-PBHNN。该方案分为四个阶段:首先,清理智能合约源代码的无用信息,保留关键语义信息;其次,将清理好的源代码序列输入到 BERT 模型训练词向量;然后,经特征提取层提取特征向量,捕捉源代码序列中的特征;最后,将获得的特征向量输入到漏洞检测层进行分类,输出检测结果。实验结果表明,与现有的智能合约漏洞检测方法相比,该方案在检测四类漏洞准确率方面均达到 90% 以上。下一步工作将采用更轻量级的动态预训练语言模型进行词义特征提取,进一步降低训练成本并提高漏洞检测的准确率。

参考文献:

[1]　WU H, DONG H, HE Y, et al. Smart contract vulnerability

detection based on hybrid attention mechanism model[J]. Applied Sciences,2023,13(2):770.
[2] ZHANG L,WANG J,WANG W,et al. Smart contract vulnerability detection combined with multi-objective detection [J]. Computer Networks,2022,217:109289.
[3] DENG W,WEI H,HUANG T,et al. Smart contract vulnerability detection based on deep learning and multimodal decision fusion[J]. Sensors,2023,23(16):7246.
[4] RAJ A,MAJI K,SHETTY S D. Ethereum for Internet of Things security[J]. Multimedia Tools and Applications,2021,80(12):18901-18915.
[5] 王雅静. 面向以太坊智能合约的安全漏洞检测方法研究与实现[D]. 北京:北京工业大学,2022.
[6] 白英民. 基于时序特征和文本分类的智能合约漏洞检测方法研究[D]. 太原:中北大学,2023.
[7] GARFATTA I,KLAI K,GAALOUL W,et al. A survey on formal verification for solidity smart contracts[C]//Proceedings of the 2021 Australasian computer science week multiconference. Dunedin: Association for Computing Machinery,2021:1-10.
[8] 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. [s. l.]:Association for Computing Machinery,2016:254-269.
[9] REN X,WU Y,LI J,et al. Smart contract vulnerability detection based on a semantic code structure and a self-designed neural network[J]. Computers and Electrical Engineering,2023,109:108766.
[10] 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. Montpellier: Association for Computing Machinery,2018:259-269.
[11] NGUYEN T D,PHAM L H,SUN J,et al. Sfuzz:an efficient adaptive fuzzer for solidity smart contracts[C]//Proceedings of the ACM/IEEE 42nd international conference on software engineering. Seoul: Association for Computing Machinery,2020:778-788.
[12] HILDENBRANDT E,SAXENA M,RODRIGUES N,et al. Kevm:a complete formal semantics of the ethereum virtual machine[C]//2018 IEEE 31st computer security foundations symposium (CSF). Oxford:IEEE,2018:204-217.
[13] 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. Gothenburg:Association for Computing Machinery,2018:9-16.
[14] 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). Montreal:IEEE,2019:8-15.
[15] SO S,LEE M,PARK J,et al. Verismart:a highly precise safety verifier for ethereum smart contracts[C]//2020 IEEE symposium on security and privacy (SP). San Francisco:IEEE,2020:1678-1694.
[16] DEMERTZIS K,ILIADIS L,TZIRITAS N,et al. Anomaly detection via blockchained deep learning smart contracts in industry 4.0[J]. Neural Computing and Applications,2020,32(23):17361-17378.
[17] 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. Yokohama:IJCAI,2021:3283-3290.
[18] WANG W,SONG J,XU G,et al. Contractward:automated vulnerability detection models for ethereum smart contracts [J]. IEEE Transactions on Network Science and Engineering,2020,8(2):1133-1144.
[19] YU X,ZHAO H,HOU B,et al. Deescvhunter:a deep learning-based framework for smart contract vulnerability detection[C]//2021 international joint conference on neural networks (IJCNN). Shenzhen:IEEE,2021:1-8.
[20] 孟祥爱. 面向以太坊智能合约漏洞的高效检测算法研究[D]. 北京:北京交通大学,2022.
[21] JEON S,LEE G,KIM H,et al. Smartcondetect:highly accurate smart contract code vulnerability detection mechanism using bert[C]//KDD workshop on programming language processing (PLP). Singapore:ACM,2021:1-8.
[22] 李 鑫,杜景林,陈子文,等. 基于双通道的智能合约漏洞检测方法[J]. 科学技术与工程,2023,23(34):14651-14659.
[23] 林彦君,张 龑. 基于语义与结构特征融合的整数溢出漏洞检测[J]. 湖北大学学报:自然科学版,2024,46(4):531-539.
[24] GRAVES A,GRAVES A. Long short-term memory[C]//Supervised sequence labelling with recurrent neural networks. [s. l.]:[s. n.],2012:37-45.
[25] ULLAH A,AHMAD J,MUHAMMAD K,et al. Action recognition in video sequences using deep bi-directional LSTM with CNN features[J]. IEEE Access,2017,6:1155-1166.
[26] 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 ACM/IEEE 42nd international conference on software engineering. Seoul:Association for Computing Machinery,2020:530-541.
[27] ZHANG L,CHEN W,WANG W,et al. Cbgru:a detection method of smart contract vulnerability based on a hybrid model[J]. Sensors,2022,22(9):3577.