文章编号:1007-5321(2025)05-0055-07 DOI:10.13190/j.jbupt.2024-197

基于专家模式和可解释机器学习的智能合约漏洞检测

谈 聪, 李 彪, 李文敏, 秦素娟, 高 飞
(北京邮电大学 网络空间安全学院, 北京 100876)

摘要: 智能合约是运行在区块链上的一段计算机程序,具有自动执行、不可篡改、公开等特性。智能合约控制大量高价值数据的流动,攻击者可以利用智能合约存在的漏洞窃取资金或资源。现有的检测方法,如符号执行存在路径爆炸、误报率较高等问题,机器学习方法是黑盒的,有不可解释性。针对上述问题,提出了基于专家模式和可解释的机器学习来进行智能合约代码漏洞的检测,设计漏洞的专家模式,使用可解释性机器学习(SHAP)来解释多种特征的权重,针对 4 种漏洞(重入漏洞、时间戳漏洞、整数溢出漏洞、权限控制漏洞)的平均检测准确率达到了 90.36%,和 Oyente、Mythril 等经典工具相比取得了更好的检测效果。

关 键 词: 区块链; 智能合约; 漏洞检测; 机器学习; 专家模式
中图分类号: TP391.1 **文献标志码**: A

Smart Contract Vulnerability Detection Based on Expert Pattern and Explainable Machine Learning

TAN Cong, LI Biao, LI Wenmin, QIN Sujuan, GAO Fei
(School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: A smart contract is a piece of computer program that runs on the blockchain, which has the characteristics of automatic execution, non-tampering, and publicity. Smart contracts control the flow of large amounts of high-value data, and attackers can exploit vulnerabilities in smart contracts to steal funds or resources. Existing detection methods, such as symbol execution, have problems such as path explosion and high false positive rate, while machine learning methods are black-box and uninterpretable. In order to solve the above problems, an expert mode based on expert mode and explainable machine learning was proposed to detect vulnerabilities in smart contract code, an expert mode for vulnerabilities was designed, and shapley additive explanations (SHAP) was used to explain the weights of multiple features, and the average detection accuracy of four vulnerabilities (re-entrancy vulnerability, timestamp vulnerability, integer overflow vulnerability, and permission control vulnerability) reached 90.36%, which achieved better detection results compared with classic tools such as Oyente and Mythril.

Key words: blockchain; smart contract; vulnerability detection; machine learning; expert pattern

以太坊由 Vitalik Buterin 于 2015 年推出,是一个去中心化的开源区块链平台[1],对数字资产和去中心化应用程序的世界产生了重大影响。智能合约是以太坊上自动执行的合约,其条款直接写入代

收稿日期: 2024-10-09
基金项目: 国家重点研发计划资助项目(2021YFB2700400)
作者简介: 谈 聪(2001—), 女, 硕士生。
通信作者: 秦素娟(1979—), 女, 教授, 博士生导师, 邮箱:qsujuan@bupt.edu.cn。

码[2]。它们存储并复制到区块链网络上,以确保透明度、可验证性和不变性。一旦满足指定的条件,智能合约就会自动执行商定的操作,而无需中介。关注以太坊上智能合约的安全性至关重要,因为任何违规行为都可能导致重大财务损失[3],并破坏对生态系统的信任。

智能合约作为可信的去中心化应用,获得了广泛关注,但是其存在的安全漏洞问题也给智能合约的可靠性带来了巨大的威胁。智能合约上链后若存在漏洞无法被修复,这对智能合约的可靠性提出了更高的要求。目前,研究者们使用多种方法,比如符号执行、形式化验证、中间表示、模糊测试、深度学习等来进行智能合约的漏洞检测[4],取得了一定的成果。

符号执行方法先将合约中的变量值符号化,在逐条解释执行程序中指令的过程中,更新执行状态,搜集路径约束,以完成程序中所有可执行路径的探索,并发现相应的安全问题,但存在路径爆炸与约束求解难[5]、误报漏报率较高等问题。

形式化验证是最严谨的智能合约安全验证技术,但其自动化程度相对较低,且无法动态分析,缺乏对漏洞检测结果的可达性检验,会产生较高的误报率。形式化验证可能无法应对智能合约中的所有情况,因此可能存在遗漏的情况。

模糊测试是当前流行的漏洞检测技术之一,从目标应用程序中生成大量正常和异常的测试用例,尝试将生成的用例提供给目标应用程序,并监视执行状态中的异常结果以发现安全问题,可发现智能合约中的未预料到的输入和边界情况。但是通常需要大量的随机输入来覆盖各种情况,可能无法覆盖所有可能的路径和漏洞。

近年来,深度学习在代码漏洞检测领域已经有越来越多的成功实践,在智能合约漏洞检测领域也有一定程度的应用,与传统漏洞挖掘方法相比,引入深度技术可缓解传统方法耗费的大量人力与误报率、漏报率较高的问题,但是目前深度学习在检测智能合约跨函数漏洞上的检测效果不佳[4]。另外,深度学习方法依赖数据集、算法,存在源码语义建模不足和检测结果可解释性较差[6]等问题。

为了缓解机器学习检测智能合约漏洞的黑盒问题,以及应对机器学习检测合约中跨函数漏洞准确率低的问题,基于开源的智能合约数据集,结合设计的专家模式使用机器学习的模型来进行4种类型:重入漏洞、时间戳漏洞、整数溢出漏洞、权限控制漏洞的代码漏洞检测。4种漏洞的平均检测准确率、误报率、漏报率分别为90.36%,14.18%,14.04%,显著提升了漏洞检测效果。对提取出来的多种特征进行了权重解释,通过比较多种特征对于模型训练的贡献程度来为机器学习提供解释性。

1 智能合约

1.1 重入漏洞

智能合约的重要特点之一就是调用和利用其他外部的合约。通常在合约中会发生一些将以太币发送给外部用户账户的行为,而转账和调用外部合约的操作需要智能合约的外部调用,若这些外部调用操作不慎,则极其容易被攻击者利用。可重入性通常表现为一个函数的同时多次调用,恶意合约在调用其他函数完成之前多次调用被攻击函数,在被攻击合约中“重新输入”了代码执行[7],从而实现攻击,这可能会造成巨大的破坏。

1.2 整数溢出漏洞

当算术运算达到类型的最大或最小值时,将发生上溢或者下溢。例如,如果一个数字以uint8类型存储,则意味着该数字以8位无符号数字存储,值范围从0到2^8-1。当算术运算试图创建一个超出指定位数表示范围的数值时,就会发生整数溢出。在Solidity语言中也会出现类似的问题。

1.3 时间戳漏洞

合约通常需要访问时间值以执行某种功能。block.timestamp和block.number可以使开发者了解当前区块的时间参数,但是大多数情况下使用它们并不安全。对于block.timestamp,开发人员经常尝试使用它来触发时间相关的事件。由于以太坊是去中心化的节点,只能在某种程度上同步时间[8]。此外,恶意矿工可以更改其区块的时间戳。

1.4 权限控制漏洞

Solidity语言中函数和变量的4种可见性修饰包含:public,internal,private,external。如果开发人员未设置函数或状态变量的可见性,或可见性为public或internal,并且恶意用户能够进行未经授权或意外的状态更改,则可能导致漏洞。

2 基于专家模式和可解释机器学习的智能合约漏洞检测方案

针对深度学习在检测智能合约跨函数漏洞上检

测效果不佳、源代码语义建模不足、检测结果可解释性差的问题，提出了基于专家模式和可解释机器学习的智能合约漏洞检测方案。

方案检测流程如下：首先对智能合约 Solidity 源代码进行预处理，提取出源代码的语义特征和操作码的结构特征，使用漏洞专家知识经跨函数静态分析模块分析得到专家模式特征，输入特征到深度学习模型中训练，使用可解释性机器学习（SHAP，shapley additive explanations）库进行可解释的特征权重计算，特征筛选模块利用深度学习解释得到的专家模式特征、源代码语义特征、操作码序列结构特征的权重，上述权重可以展示不同特征在模型训练过程中起到的作用，最后输出漏洞检测报告。图 1 为笔者方案的检测智能合约代码漏洞的流程图。

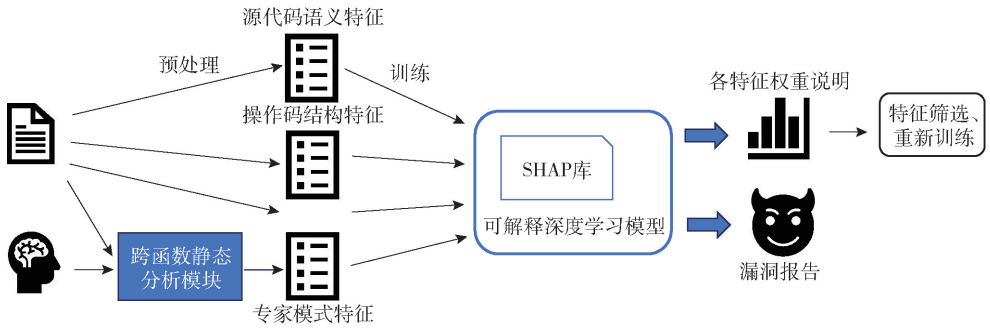

图 1　检测智能合约漏洞流程图

2.1　数据预处理

对源代码进行预处理，包含去除 Solidity 代码中的注释、换行符、空格等。在作数据预处理的时候，可以捕捉与漏洞相关的重要变量所在的源代码片段，去除源代码中的噪音代码。如未对 msg. sender 进行正确的权限验证，可能导致未经授权的地址执行合约中的敏感操作；不正确地使用 block. timestamp 可能导致时间依赖性的漏洞，如时间戳漏洞；不正确地使用合约自身的地址可能导致重入攻击或其他安全问题。

源代码编译得到字节码，处理字节码得到合约中的控制流图（CFG，control flow graph），针对 CFG 进行深度优先搜索（DFS，depth-first search）遍历得到操作码路径。智能合约中与漏洞密切相关的操作码[9]包含获取当前合约地址的 address、获取指定地址的以太币余额的 balance、根据权限条件进行跳转或回滚的操作码 jump 和 jumpi、存储和读取状态变量的 sstore 和 sload 等。可以结合与漏洞相关的操作码对操作码路径进行去噪处理。

2.2　特征提取

提取源代码语义特征、操作码结构特征、专家模式特征作为机器模型训练的输入。

1）源代码语义特征提取。将经过预处理后的源代码划分为 token；然后使用自然语言处理的方法从中提取出智能合约的语义特征。使用预训练的词嵌入模型 Word2Vec[10]将 token 转换为密集的向量表示；使用 n-gram 模型来捕捉标记序列的局部上下文信息，使用 3-gram 来提取相邻标记的组合特征。

2）操作码结构特征提取。操作码特征指的是操作码序列包含的结构特征，针对智能合约源代码使用 solc 编译器编译得到字节码，然后使用 evm_cfg_builder 生成字节码对应的 CFG 图文件，使用 DFS 算法遍历 CFG 图得到操作码序列的路径，路径中即包含了智能合约代码的结构信息。

3）专家模式特征提取。根据智能合约漏洞库漏洞原理获取专家知识，使用跨函数静态分析模块分析敏感路径，在敏感路径上进行专家知识校验，得到漏洞的专家模式特征。

跨函数静态分析模块流程如下：构建 CFG 和数据依赖图（DDG，data dependency graph），然后将 CFG 和 DDG 合并成一个综合图，这样可以同时考虑控制流和数据流。在综合图中标记所有涉及到权限控制和敏感操作的关键节点。权限控制节点通常是指那些包含权限检查的地方，如函数调用或状态变量修改；而敏感操作节点是指那些执行关键动作的地方，比如转移资产或修改重要状态变量。从权限控制节点出发，寻找所有通往敏感操作节点的路径，并构建路径集合。

以重入漏洞为例，通过在跨函数静态分析模块收集到的路径上实现算法 1，提取专家模式的算法如下所示。

算法 1　提取重入漏洞专家模式

输入:contract 智能合约源代码
中间: allFunctionList 所有的函数存放在列表
callValueFunctionList 调用了 call. value 的函数存放在列表里
otherFunctionList 其他函数存放在列表里
输出:patterns 提取出的重入漏洞专家模式
Begin
Patterns 初始化为空列表。
对 contract 进行预处理:去除注释空格等。
切割得到 contract 中的函数列表 allFunctionList:
for function in allFunctionList
  if‘call. value’ in function
    callValueFunctionList. add(function)
  else
    otherFunctionList. add(function)
  if len(callValueFunctionList)! =0
    patterns. append(1)
    if callValueFunctionList 调用了 call. value 的变量进行算术操作:
      patterns. append(1)
      if call. value 的参数在后面出现:
        patterns. append(1)
  else
    patterns. append(0)
End

2.3 模型训练与漏洞检测

将 2.2 节中提取出的 3 种特征放到机器学习模型中训练,并使用 SHAP 库来对权重进行解释。SHAP 是一种流行的机器学习模型解释性工具,它基于博弈论中的 Shapley 值来解释模型的预测结果。SHAP 的核心思想是,模型的每个预测都是由多个特征的“贡献”组成的,这些贡献可以被量化为 SHAP 值[11]。通过计算每个特征对预测结果的 SHAP 值,可以了解每个特征对模型输出的影响程度。

输入特征到深度学习的模型中预训练完成后,使用 SHAP 算法来评估模型的预测。下面为使用 SHAP 库解释多种特征的算法 2。

算法 2 SHAP 实现特征权重的可解释性
输入:model 预训练好的机器学习模型;data 特征数据集;n 特征数量;m 数据集中样本的数量
输出:shap_values 包含每个特征的 SHAP 值数组
Begin
初始化 shap_values 数组,长度为 m,存储每个样本的 SHAP 值。
计算模型预测的基线值 base_value:
  base_value = model(data)
  for i in data
    特征 i 的权重初始化:$P_i=0$
    对每个特征子集 S except i
      计算子集 S 的权重 w
      计算子集 S 与特征 i 结合的模型预测值 F_i = model(data[$S\cup\{i\}$])
      计算子集 S 的模型预测值 F = model(data[S])
      累加特征 i 的增量 $P_i=P_i+w(F_i-F)$
    sharp_values[i] = P_i
  return sharp_values
End

3 实验结果与分析

3.1 实验环境

使用的实验条件如下:配置为 4 核、8 GB 内存、8 Mbit/s 带宽的 Ubuntu Server 20.04 长期支持(LTS, long term support)64 位服务器。服务器的系统盘容量为 500 GB。代码开发工具为 Visual Studio Code。使用的 Python 环境版本为 3.7.2。

笔者从已发布的文献和 Awesome-Buggy-ERC20-Tokens[12]项目收集智能合约,最终共收集了 51 039 个智能合约,去除重复的合约和空白的合约后,共包含 18 131 个合约。其中,包含漏洞的合约有 8 523 个,包含重入漏洞的合约有 2 392 个,包含权限控制漏洞的合约有 1 596 个,包含整数溢出漏洞的合约有 3 419 个,包含时间戳漏洞的合约有 1 116 个,不包含漏洞的合约有 9 608 个,数据分布如图 2 所示。

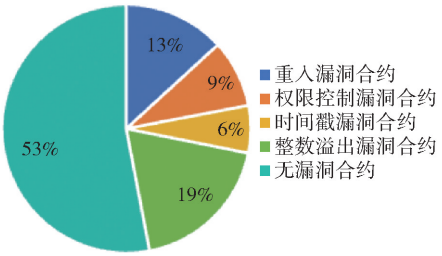

图 2 数据集中各漏洞合约占比

3.2 实验指标

实验评价指标包含精确率、漏报率、误报率。

1）精确率：表示检测准确的个数占所有样本的比例。

2）漏报率：表示在检测没有漏洞但实际上包含漏洞的合约个数占实际上所有包含漏洞合约个数的比例，也就是漏报合约的个数占所有具有漏洞合约个数的比例。

3）误报率：表示在检测包含漏洞但是实际上并没有漏洞的合约个数占实际上无漏洞合约个数的比例，也就是误报合约的个数占所有正常合约个数的比例。

3.3　实验结果与分析

实验 1　为了研究笔者方案检测效果是否优于智能合约漏洞检测领域经典方法的问题，调研文献后，选择下面经典智能合约代码漏洞工具在 3.1 节的智能合约的数据集上作实验比对。

Mythril[13]是一种智能合约静态分析工具，其使用概念分析、污点分析和控制流验证来检测以太坊智能合约中的漏洞，包含重入漏洞、整数溢出漏洞、时间戳漏洞、权限控制漏洞等。

Oyente[14]以智能合约字节码和以太坊状态作为输入，模拟以太坊虚拟机（EVM，Ethereum virtual machine），并且遍历合约的不同执行路径，其支持检测的漏洞类型包括重入漏洞、整数溢出漏洞、时间戳漏洞等。

TeEther[15]是一种智能合约静态分析工具，区别于其他工具，它考虑了智能合约漏洞自动识别以及合约生成方法，并通过分析合约字节码查找关键的执行路径，以检测合约的安全问题。可检测权限控制漏洞等。

GNNSCVulDetector[16]是一种使用图神经网络进行智能合约漏洞检测的技术。可针对重入漏洞、整数溢出漏洞、时间戳漏洞、权限控制漏洞进行检测。实验结果如表 1 所示。

使用 SHAP 库对训练好的 4 种漏洞的检测模型进行权重分析的结果如图 3 所示。图 3 中 weight_expert，weight_structure，weight_code 分别表示专家模式特征、操作码结构特征、代码语义特征对于模型训练结果的贡献程度，SHAP 值越大，贡献程度越高。若某一类特征的 SHAP 值低于 5%，说明该特征对模型训练的贡献程度不大，需要进行特征筛选重新训练。

下面以重入漏洞为例分析其特征权重。比较语义特征 weight_code、结构特征 weight_structure、专家知识特征 weight_expert 大小：weight_structure > weight_expert > weight_code。这表明在重入漏洞检测中，操作码结构特征对模型的贡献度最高，其次是专家模式特征，最后是代码语义特征。这一结果与重入漏洞的原理相符，因为重入漏洞通常与递归调用转账函数有关，这是一种结构上的特征。重入漏洞的原理可简述为递归调用某个转账函数，递归调用这一动作和结构特征联系紧密，同时 weight_expert > weight_code 说明专家模式特征能够捕捉到更深层次的、与漏洞相关的模式。由于源代码的语义特征对于检测的贡献度较低，于是仅输入结构特征和专家特征重新训练机器学习的模型，发现准确率没有降低，可以筛除掉源代码的语义特征，这也验证了特征筛选的有效性。去除不必要的特征可以减少模型的复杂度，防止过拟合。

表 1　本方案与其他工具 4 种漏洞检测效果对比　%

漏洞类别	方案	精确率	误报率	漏报率
重入漏洞	Oyente	74.10	13.05	30.02
	Mythril	80.43	17.93	28.54
	GNNSCVulDetector	81.57	19.17	24.42
	笔者方案	95.17	10.85	13.78
权限控制漏洞	Mythril	77.89	19.20	24.76
	GNNSCVulDetector	67.90	33.08	26.15
	TeEher	82.15	23.73	8.94
	笔者方案	86.52	20.04	9.86
时间戳依赖漏洞	Oyente	81.04	18.15	29.23
	Mythril	83.66	23.42	28.94
	GNNSCVulDetector	79.51	24.67	19.69
	笔者方案	90.73	9.49	12.33
整数溢出漏洞	Oyente	76.74	18.96	41.73
	Mythril	78.15	15.42	35.75
	GNNSCVulDetector	70.03	21.88	15.79
	笔者方案	89.04	16.35	20.18

实验 2　为了比对笔者方案和大模型在数据集上的检测效果，使用 Meta 公司推出的大语言模型元人工智能[17]（LLaMA，large language model meta AI）进行实验对比。在 3.1 节中的数据集上使用 LLaMA 3.2 版本模型，基于提示词工程设计输入的问题检测智能合约漏洞。将大模型输出的实验结果进行统计，与论文提出方案进行对比，得到的实验结果如表 2 所示。

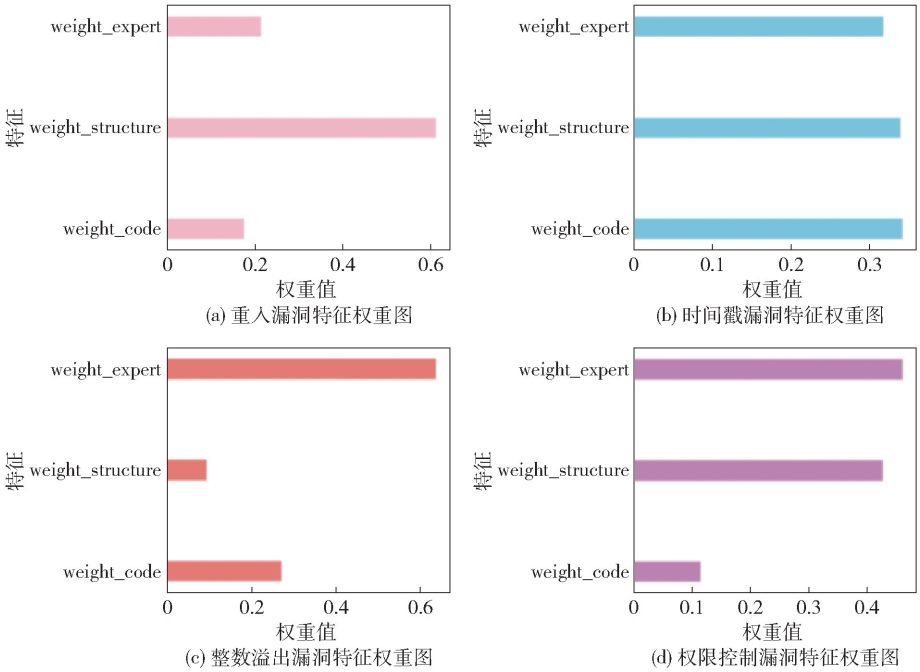

图3 各种漏洞的特征权重

表2 笔者方案与Meta公司大模型在各漏洞上的检测效果对比 %

漏洞类别	方案	精确率	误报率	漏报率
重入漏洞	大模型	68.74	21.93	33.45
	笔者方案	95.17	10.85	13.78
权限控制漏洞	大模型	83.74	2.86	21.15
	笔者方案	86.52	20.04	9.86
时间戳依赖漏洞	大模型	91.11	5.31	35.40
	笔者方案	90.73	9.49	12.33
整数溢出漏洞	大模型	50.94	12.40	52.36
	笔者方案	89.04	16.35	20.18

根据表2可知,LLaMA 3.2大模型在数据集上的时间戳漏洞检测效果比笔者方案好,但是在其他3种漏洞上的检测效果较差。实验中发现针对长合约的检测效果较差,目前LLaMA 3.2处理这种长文本的能力还不符合预期,需要结合专家知识来检测。

4 结束语

针对机器学习的难解释性和主流工具检测误报漏报问题,设计了检测工具,基于4类漏洞(重入漏洞、时间戳漏洞、整数溢出漏洞、权限控制漏洞)的原理设计专家模式,将专家模式特征和字节码中提取的特征输入到机器学习的模型中,并给出训练过程不同特征的权重分析结果,赋予工具检测结果以可解释性。从公开的漏洞数据库中收集了智能合约数据集,并进行了实验,证明了笔者工具的有效性,其准确率超过了现有的智能合约漏洞检测工具如Oyente等,降低了误报率和漏报率。同时,针对部分漏洞目前没有全面的检测工具,后续可以收集智能合约的其他漏洞类别的数据集,设计专家模式,扩充工具对于漏洞类别的覆盖面。

参考文献:

[1] CONTI M, KUMAR S, LAL C, et al. A survey on security and privacy issues of bitcoin. IEEE Communications Surveys and Tutorials, 2018, 20(4): 3416-3452.

[2] SZABO N. The idea of smart contracts. Nick Szabo's Papers and Concise Tutorials, 1997, 6(1): 199.

[3] ATZEI N, BARTOLETTI M, CIMOLI T. A survey of attacks on ethereum smart contracts (sok). In Principles of Security and Trust, 2017: 164-186.

[4] 董伟良, 刘哲, 刘逵, 等. 智能合约漏洞检测技术综述[J]. 软件学报, 2024, 35(1): 38-62.
DONG W L, LIU Z, LIU K, et al. Survey on vulnerability detection technology of smart contracts[J]. Journal of Software, 2024, 35(1): 38-62.

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

[6] 张小松, 牛伟纳, 黄世平, 等. 基于深度学习的智能合约漏洞检测方法综述[J]. 四川大学学报: 自然科

学版，2023，60：020001.
ZHANG X S，NIU W N，HUANG S P，et al. A survey of smart contract vulnerability detection methods based on deep learning[J]. Journal of Sichuan University (Natural Science Edition)，2023，60：020001.

[7] PEREZ D，LIVSHITS B. Smart contract vulnerabilities：Vulnerable does not imply exploited. In 30th USENIX Security Symposium (USENIX Security 21)，2021：1325-1341.

[8] 范吉立，李晓华，聂铁铮，等. 区块链系统中智能合约技术综述[J]. 计算机科学，2019，46(11)：1-10.
FAN J L，LI X H，NIE T Z，et al. Survey on smart contract based on blockchain system[J]. Computer Science，2019，46(11)：1-10.

[9] CHEN T，LI Z H，ZHANG Y F，et al. A large-scale empirical study on control flow identification of smart contracts. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)，2019：1-11.

[10] MIKOLOV T，CHEN K，CORRADO G，et al. Efficient estimation of word representations in vector space[J]. Computer Science. arXiv preprint arXiv 2013：1301.3781.

[11] SHAP. Welcome to the SHAP documentation[EB/OL]. 2024. https：//shap. readthedocs. io/en/latest/index. html.

[12] SEC-BIT. Awesome-buggy-erc20-tokens [EB/OL]. 2019. https // github. com/sec-bit/ awesome-buggy-erc20-tokens.

[13] ConsenSys. Mythril-security analysis tool for evm bytecode[EB/OL]. 2021. https：//github. com/ConsenSys/mythril.

[14] LUU L，CHU D，OLICKEL H，et al. Making smart contracts smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security，2016：254-269.

[15] KRUPP J，ROSSOW C. TeEther：Gnawing at ethereum to automatically exploit smart contracts. In 27th USENIX Security Symposium (USENIX Security 18)，2018：1317-1333.

[16] ZHUANG Y，LIU Z，QIAN P，et al. Smart contract vulnerability detection using graph neural network[C]// IJCAI，2020：3283-3290.

[17] TOUVRON H，LAVRIL T，IZACARD G，et al. Llama：Open and efficient foundation language models[J]. arXiv preprint arXiv 2023：2302. 13971.

(上接第 39 页)

[12] 靳喜博，刘琨，江俊峰，等. 基于卷积神经网络的多维度分布式光纤振动传感事件识别[J]. 光学学报，2024，44(1)：384-394.
JIN X B，LIU K，JIANG J F，et al. Multi-dimensional distributed optical fiber vibration sensing patterner cognition based on convolutional neural network[J]. Acta Optica Sinica，2024，44(1)：384-394.

[13] LI S，PENG R，LIU Z. A surveillance system for urban buried pipeline subject to third-party threats based on fiber optic sensing and convolutional neural network[J]. Structural Health Monitoring，2021，20(4)：1704-1715.

[14] YANG K，DING Y，GENG F，et al. A multi-sensor mapping Bi-LSTM model of bridge monitoring data based on spatial-temporal attention mechanism[J]. Measurement，2023，217：113053.

[15] ADEMUJIMI T，PRABHU V. Fusion-learning of Bayesian network models for fault diagnostics[J]. Sensors，2021，21：7633.