软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(9):4193−4217 [doi: 10.13328/j.cnki.jos.007133]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于形式化方法的区块链系统漏洞检测模型*

陈锦富[1,2], 冯乔伟[1,2], 蔡赛华[1,2], 施登洲[1,2], Rexford Nii Ayitey SOSU[1,3]

[1](江苏大学 计算机科学与通信工程学院, 江苏 镇江 212013)

[2](江苏省工业网络安全技术重点实验室 (江苏大学), 江苏 镇江 212013)

[3](Faculty of Computing and Information Systems, Ghana Communication Technology University, Accra 23321, Ghana)

通信作者: 蔡赛华, E-mail: caisaih@ujs.edu.cn

摘　要: 随着区块链技术在各行各业的广泛应用, 区块链系统的架构变得越来越复杂, 这也增加了安全问题的数量. 目前, 在区块链系统中采用了模糊测试、符号执行等传统的漏洞检测方法, 但这些技术无法有效检测出未知的漏洞. 为了提高区块链系统的安全性, 提出基于形式化方法的区块链系统漏洞检测模型 VDMBS (vulnerability detection model for blockchain systems), 所提模型综合系统迁移状态、安全规约和节点间信任关系等多种安全因素, 同时提供基于业务流程执行语言 BPEL (business process execution language) 的漏洞模型构建方法. 最后, 用 NuSMV 在基于区块链的电子投票选举系统上验证所提出的漏洞检测模型的有效性, 实验结果表明, 与现有的 5 种形式化测试工具相比, 所提出的 VDMBS 模型能够检测出更多的区块链系统业务逻辑漏洞和智能合约漏洞.

关键词: 区块链系统; 安全因素; 漏洞检测模型; 形式化验证; BPEL 流程

中图法分类号: TP311

中文引用格式: 陈锦富, 冯乔伟, 蔡赛华, 施登洲, Sosu RNA. 基于形式化方法的区块链系统漏洞检测模型. 软件学报, 2024, 35(9): 4193–4217. http://www.jos.org.cn/1000-9825/7133.htm

英文引用格式: Chen JF, Feng QW, Cai SH, Shi DZ, Sosu RNA. Vulnerability Detection Model for Blockchain Systems Based on Formal Method. Ruan Jian Xue Bao/Journal of Software, 2024, 35(9): 4193–4217 (in Chinese). http://www.jos.org.cn/1000-9825/7133.htm

Vulnerability Detection Model for Blockchain Systems Based on Formal Method

CHEN Jin-Fu[1,2], FENG Qiao-Wei[1,2], CAI Sai-Hua[1,2], SHI Deng-Zhou[1,2], Rexford Nii Ayitey SOSU[1,3]

[1](School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang 212013, China)

[2](Jiangsu Key Laboratory of Security Technology for Industrial Cyberspace (Jiangsu University), Zhenjiang 212013, China)

[3](Faculty of Computing and Information Systems, Ghana Communication Technology University, Accra 23321, Ghana)

Abstract: As blockchain technology is widely employed in all walks of life, the architecture of blockchain systems becomes increasingly more complex, which raises the number of security issues. At present, traditional vulnerability detection methods such as fuzz testing and symbol execution are adopted in blockchain systems, but these techniques cannot detect unknown vulnerabilities effectively. To improve the security of blockchain systems, this study proposes a vulnerability detection model for blockchain systems (VDMBS) based on the formal method. This model integrates multiple security factors including system migration state, security property and trust relationship among nodes, and provides a vulnerability model building method based on business process execution language (BPEL). Finally, the effectiveness of the proposed vulnerability detection model is verified on a blockchain-based e-voting election system by NuSMV, and the experimental results show that compared with five existing formal testing tools, the proposed VDMBS model can detect more blockchain system logic vulnerabilities and smart contract vulnerabilities.

* 基金项目: 国家重点研发计划 (2020YFB1005501); 国家自然科学基金 (62172194, 62202206, U1836116); 江苏省自然科学基金 (BK20220515); 中国博士后科学基金 (2023T160275); 江苏省自然科学基金前沿技术项目 (BK20202001); 江苏省青蓝工程
本文由“形式化方法与应用”专题特约编辑曹钦翔副教授、宋富研究员、詹乃军研究员推荐.
收稿时间: 2023-09-11; 修改时间: 2023-10-30; 采用时间: 2023-12-13; jos 在线出版时间: 2024-01-05
CNKI 网络首发时间: 2024-03-29

© 中国科学院软件研究所 http://www.jos.org.cn

Key words: blockchain system; security factor; vulnerability detection model; formal verification; BPEL flow

作为一种全新的分布式数据存储技术, 区块链以其易管理、可扩展的特点被广泛应用于各个领域中. 相较于传统软件系统, 区块链系统以其去中心化、不可篡改的特性, 为用户提供了安全、高效等多重便利. 然而, 与传统软件类似, 分布式计算的复杂性也给区块链技术的发展带来了挑战, 这些挑战包括但不限于设计、实现、评估和安全维护等多个方面[1−3]. 目前使用的区块链系统中包含各种未知或已知的漏洞, 攻击者通过这些现有的漏洞 (如 51% 攻击、拒绝服务攻击等) 对区块链系统构成极大的安全风险[4,5]. 因此, 要确保区块链系统的可靠性和安全性则必须对其潜在漏洞进行全面的检测和分析, 从而保障其正常、安全的运行.

相较于传统的软件系统, 对区块链系统的漏洞检测研究应该考虑到区块链系统中应用层、合约层、网络层、共识层以及数据层多个层面的因素[6]. 此外, 针对不同的层面, 测试技术的要求也呈现出多样性. 例如, 在应用层测试中广泛采用渗透测试, 但是安全测试人员的疏忽或者经验不足导致漏洞不易被检测[7]. 在区块链系统的实际应用中, 与智能合约相关的安全事件占据了相当大的比例, 而开发者缺乏足够的经验很容易导致漏洞的出现, 这也是区块链系统安全测试研究的主要关注点. 智能合约漏洞检测的常见方法包括模糊测试和符号执行等[8]. 在进行模糊测试时由于随机种子难以覆盖所有路径, 因此检测精度并不理想[9]. 对于符号执行, 它通过覆盖所有路径和测试每个可能的程序执行路径来检测漏洞, 这种方法能够较好地发现隐藏于代码中的错误, 提高了程序的安全性; 然而, 随着被测区块链系统复杂度的不断增加, 所需的测试时间也随之显著增加[10,11]. 在对网络层进行安全测试时, 常用的方法或工具包括入侵检测[12]、防火墙[13]、扫描仪[14]等, 但是都无法有效地检测出潜在的漏洞. 此外, 在共识层和数据层使用的安全测试方法也存在着人工干预多、缺乏普遍性、无法及时发现区块链系统运行过程中潜在漏洞的缺点[15].

虽然近年来针对区块链系统的漏洞检测取得了一定的进展, 但对于层出不穷的面向区块链的漏洞攻击手段以及缺乏针对区块链系统的漏洞检测方法, 现有方法仍然难以有效检测区块链系统中的未知漏洞. 此外, 区块链系统的大规模、高维护要求以及独特的分布式特点使得很难对其进行直接的分析, 而传统的安全检测方法也无法对区块链系统的安全策略和漏洞进行统一的全面分析. 与之相比, 形式化方法可以抽象出系统的本质过程, 简化了分析过程, 从而全面分析区块链系统中各层、各环节的安全性.

当前, 利用形式化理论对整个区块链系统安全性进行分析的研究较少, 大多集中于对区块链系统中共识协议和智能合约的形式化验证. 例如, Afzaal 等人[16]利用模型检查技术对区块链众包系统中安全可信众包共识协议进行形式化验证, 确保该共识协议的正确性. Ribeiro 等人[17]使用 Isabelle/Hol 工具对 Solidity 智能合约进行形式化验证, 并检测出整型溢出漏洞和重入漏洞这两种漏洞. 目前, 针对区块链系统中共识协议和智能合约的形式化验证研究, 只能分析单一模块或者组件的安全性, 形式化模型简单且不完整, 不能综合考虑影响整个区块链系统运行过程中安全因素, 从而不能全面统一地对区块链系统进行安全性的分析. 因此, 在充分了解区块链系统安全漏洞的基础上, 有必要构建一个完整的漏洞检测模型, 综合考虑区块链系统中的安全因素, 为基于形式化方法的区块链系统安全分析提供理论指导.

针对上述问题, 本文提出了一种基于形式化方法的区块链系统漏洞检测模型 VDMBS (vulnerability detection model for blockchain systems), 主要贡献包括: (1) 集成了影响区块链系统安全的多重因素 (如状态、行为、节点、功能、安全属性和模块交互等), 并引入了形式化理论对区块链系统漏洞检测模型进行形式化定义, 同时在业务流程执行语言 (BPEL) 基础上构建区块链系统漏洞检测模型. (2) 通过对一个区块链电子投票系统的实例分析, 验证漏洞检测模型构建方法的可行性, 同时利用 NuSMV 工具对所构建的安全属性规约进行验证, 证明区块链漏洞检测模型在区块链安全测试中的有效性. (3) 对 4 种不同类型的区块链系统进行形式化建模和验证, 从而证明了所提出的 VDMBS 模型在不同类型区块链系统上的普遍适用性, 以及其在检测系统交互逻辑和网络攻击方面的良好效果. (4) 将提出的 VDMBS 模型与现有的区块链系统形式化工具进行了对比分析, 证明 VDMBS 在检测系统逻辑漏洞和网络攻击风险方面的具有一定的优势.

本文第 1 节介绍区块链系统漏洞检测相关方法和研究现状. 第 2 节介绍本文所需的形式化理论的基础知识.

© 中国科学院软件研究所 http://www.jos.org.cn

第 3 节介绍本文提出的基于形式化方法的区块链系统漏洞检测模型及构建方法. 第 4 节通过实例分析验证模型的有效性. 第 5 节通过对比实验验证了模型的有效性. 第 6 节总结了全文.

1 区块链系统漏洞检测相关工作

目前, 区块链系统漏洞检测是指通过对区块链系统组件 (例如智能合约、共识算法、网络通信等) 进行分析和评估, 发现其中可能存在的漏洞和安全风险的过程, 从而确保区块链系统的安全性和可信性. 然而, 对于具有复杂结构与去中心化特征的区块链系统而言, 缺少面向该类系统的统一且全面的漏洞检测技术, 加之区块链技术及相关工具演化和改进十分迅速, 新的漏洞和安全问题也会随之产生. 近些年, 研究人员致力于研发新的区块链系统漏洞检测技术和工具来提高区块链系统漏洞检测的效率和准确率.

区块链系统漏洞检测大多集中于智能合约的漏洞, 检测漏洞的方法可以包括静态分析、动态分析、模糊测试、形式化验证、数据分析、中间表示法和深度学习等技术手段. Grishchenko 等人[18]基于形式化方法提出了 F*框架, 该方法通过将智能合约源码和字节码转化成函数编程语言 F*, 并分析和验证智能合约的安全性和功能正确性, 从而成功检测以太坊智能合约漏洞, 但是这种方法缺乏对漏洞检测结果的可达性检验, 会产生较高的误报率. Krupp 等人[19]基于符号执行技术提出了一种智能合约静态分析工具 TeEther, 考虑了智能合约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, 从而成功检测智能合约中可重入漏洞、未知函数调用漏洞以及以太冻结漏洞, 但是该方法存在路径爆炸的问题. Feist 等人[20]基于对合约的中间表示分析提出了一种智能合约静态分析框架 Slither, 该方法将智能合约 Solidity 源代码转换为 SlithIR 的中间表示, 不仅能用于检测智能合约的常见漏洞, 并且能给出合约代码优化的建议. Jiang 等人[21]基于模糊测试提出了以太坊智能合约安全漏洞的动态分析方法 ContractFuzzer, 该方法基于智能合约 ABI 规范生成模糊测试用例, 记录智能合约运行时的行为, 并能通过分析日志检测漏洞, 但是该方法存在测试用例生成低效、漏洞定位复杂等缺陷. Wang 等人[22]基于多种机器学习算法和采样算法提出了智能合约漏洞检测方法 ContractWard, 检测出了 6 种智能合约漏洞, 但是这种方法存在对智能合约源码建模不足以及检测结果可解释性较差的缺点. 总之, 现有的针对智能合约的区块链系统漏洞检测方法存在误报率高、漏洞定位复杂、漏洞类型单一等缺点.

近年来, 建立有效的漏洞检测模型来确保区块链系统的安全性正逐渐成为一个热门领域, 研究人员提出了一些针对区块链安全的漏洞分析模型. Norta 等人[23]利用 Petri 网对区块链认证协议进行安全分析, 从而发现风险和安全缺陷. Matsuo[24]根据安全需求提出了面向区块链系统智能合约的安全模型, 但该模型尚未得到实践验证. Chaudhary 等人[25]定义了一个基于双花攻击形式的模型来保证区块链共识协议的安全性. Kalra 等人[26]提出了一种基于定理证明技术的智能合约安全验证工具, 对智能合约使用符号模型检查和抽象解释进行严格证明. 然而, 这些漏洞模型分析方法只能分析区块链系统某些核心模块中的漏洞, 没有考虑区块链系统的整体以及影响区块链系统安全的各个因素. 如表 1 所示, 针对上述现有的漏洞检测模型, 对本文所提出区块链系统漏洞检测模型优势进行了分析.

表 1 区块链系统漏洞检测模型与现有漏洞检测模型

漏洞检测模型	此类漏洞检测模型的缺点	相比于此类方法区块链系统漏洞检测模型的优势
基于Petri网	仅限于认证协议	可以检测出更多类型的漏洞, 覆盖范围更广
基于安全需求	更侧重于智能合约层	更全面的安全性建模, 集成多种安全因素
基于双花攻击	只关注双花攻击	覆盖面更广, 可以检测除共识机制以外的更多层面的漏洞
基于定理证明	只能检测少量的业务逻辑漏洞	可以检测出更多业务逻辑方面的漏洞
基于状态机	只能涵盖少量的安全因素	集成了更多安全因素

如表 2 所示, 综合上述对现有漏洞检测模型的比较, 对比于现有的区块链系统漏洞检测方法, 对区块链系统漏洞检测模型的优势进行了分析.

因此, 有必要通过对各模块进行综合分析以构建区块链系统的漏洞检测模型, 从而保障区块链系统的安全.

© 中国科学院软件研究所 http://www.jos.org.cn

表 2 区块链系统漏洞检测模型与现有区块链系统漏洞检测方法

区块链系统漏洞检测方法	对比的漏洞检测方法类型	相比于此类方法区块链系统漏洞检测模型的优势
智能合约漏洞检测方法	传统智能合约漏洞检测方法	只针对智能合约漏洞, 漏洞类型单一
	区块链系统漏洞检测模型	区块链系统漏洞检测模型不仅能够检测智能合约中的漏洞, 还能检测出系统中存在的业务逻辑漏洞
其他漏洞检测方法	传统其他漏洞检测方法	只能分析区块链系统某些核心模块的安全性
	区块链系统漏洞检测模型	区块链系统漏洞检测模型考虑区块链系统的整体以及影响区块链系统安全的多种因素

2 基础知识

本文所提方法主要基于形式化方法, 下面就相关概念和基本知识进行介绍.

形式化方法 (formal method) 是一种在计算机科学和软件工程领域中广泛使用的精确建模和分析技术. 它利用数学和逻辑的原理, 以形式化的方式描述和验证计算系统的行为和性质. 形式化方法通过使用形式化规约语言 (formal specification language) 和形式化验证 (formal verification) 来对系统进行建模和分析. 它提供了一种严格、精确的描述方式, 可以清晰地定义系统的语义、规则和约束. 通过形式化方法, 开发人员可以将系统的需求和设计转化为数学模型, 然后利用形式化验证对这些模型进行验证和分析, 从而提高系统的可靠性、安全性和正确性.

2.1 形式化规约语言

形式化规约语言是一种用于描述和定义系统行为和约束的语言, 它提供了一种精确、形式化的方式来规范系统的行为. 这些规约语言使用严格的语法和语义规则, 允许开发人员明确定义系统的规则和约束, 从而帮助确保系统的正确性和一致性.

以下是几种常见的形式化规约语言.

(1) 时序逻辑 (temporal logic)

时序逻辑是一种形式化规约语言, 用于描述与系统时间相关的性质和约束. 它允许开发人员定义系统中事件和状态之间的顺序关系, 例如事件发生的先后顺序、并发行为和时间约束等. 时序逻辑常用于模型检测和形式化验证中, 检查系统是否满足特定的时序性质. 根据其特点和表达能力, 可以分为线性时序逻辑 (linear temporal logic, LTL)[27]和分支时序逻辑 (computing tree logic, CTL)[28].

(2) 过程代数 (process algebra)

过程代数[29]是一种形式化规约语言, 用于描述并发系统的行为和交互. 它提供了一组操作和规则, 用于描述并发进程之间的通信、同步和互操作. 通过过程代数, 开发人员可以建立系统模型并定义系统的行为规则, 从而便于分析和验证系统的性质和行为.

(3) 状态机 (state machines)

状态机[30]是一种形式化模型, 用于描述系统的状态和状态转换. 它由一组状态和转换规则组成, 其中状态表示系统的不同情况, 转换规则定义状态之间的转换条件和行为. 状态机可以用于建模和描述系统的行为, 以及验证系统是否满足特定的性质和约束.

有限状态自动状态机模型的形式化定义为 $M=(Q, \Sigma, \delta, q_0, F)$, 其中,

1) Q 表示的是该有限自动状态机所有状态的集合.

2) Σ 表示的是所有的输入状态的集合.

3) δ: $Q\times\Sigma\rightarrow Q$ 描述的函数关系式即从某一个状态转移到另一个状态时所遵循的规则.

4) q_0 是开始状态, 即该有限自动状态机未处理输入的时候的状态.

5) F 是终止状态, 无论一个状态机经过了怎样的变换, 最后的一部转移必须转移到接受状态, 否则就是非法的.

这些形式化规约语言提供了丰富的语法和语义, 使开发人员能够以精确和清晰的方式描述系统的行为和约

© 中国科学院软件研究所 http://www.jos.org.cn

束. 它们在形式化方法和形式化验证中扮演着重要的角色, 帮助开发人员建立可靠、正确的系统模型, 从而对系统的行为进行分析和验证. 因此, 国内外研究人员开始开展一些使用形式化规约语言对区块链系统进行形式化描述并进行安全性证明的工作.

Grishchenko 等人[18]使用形式化规约语言 F*对以太坊指令、外部调用完整性、原子性、可变账户独立性和交易环境独立性进行了形式化描述并且进行了安全性证明. Kalra 等人[26]定义了一种抽象语言来捕捉智能合约 Solidity 代码的相关构造, 并借助污点分析构造合约正确性与公平性的形式化规范, 以验证合约的正确性与公平性. Grossman 等人[31]使用 SMAC 语言形式化描述智能合约模型以及合约属性和行为, 并对通用客户端进行形式化定义, 然后对有效回归属性进行了分析. Schneidewind 等人[32]基于以太坊虚拟机字节码语义抽象解释出的霍恩子句, 并在抽象解释出的霍恩子句层面形式化描述了智能合约的行为属性, 然后实现了一个基于霍恩子句的分析框架 HoRSt, 并依据霍恩子句的数学规范在 HoRSt 上实现了分析工具 eThor.

2.2 形式化验证

形式化验证是一种通过数学和形式化方法来验证计算系统的正确性、安全性和一致性的技术. 它通过将系统的行为和性质形式化地表达为逻辑公式或规约, 然后利用推理、模型检测、定理证明等技术来自动或半自动地检查系统是否满足这些性质. 常用的形式化验证方法有模型检测 (model checking)[33]、定理证明 (theorem proving)[34]、符号执行 (symbolic execution)[35].

(1) 模型检测 (model checking)

模型检测是一种自动化的形式化验证方法, 通过对系统模型进行状态空间遍历来验证系统是否满足特定的性质. 模型检测工具会自动遍历系统的所有可能状态, 检查是否存在违反性质的状态. 它通常基于状态机模型或时序逻辑来进行验证. 模型检测方法常用的工具有 SPIN、NuSMV、UPPAL、PRISM、CADP 等[36].

(2) 定理证明 (theorem proving)

定理证明是一种基于数学推理的形式化验证方法, 通过构建逻辑证明来验证系统是否满足特定的性质. 定理证明工具使用逻辑规则和推理策略, 将系统性质转化为逻辑公式, 并尝试证明或推导出这些公式. 如果能够成功证明系统的性质, 那么该性质在给定的假设下是成立的.

(3) 符号执行 (symbolic execution)

符号执行是一种动态分析方法, 将系统的执行路径符号化, 用符号变量表示输入和状态, 从而在系统的不同路径上自动地探索和验证性质. 符号执行通过符号化的执行路径进行约束求解, 确定是否存在违反性质的路径.

形式化方法一个很大的优点就是可以检测出系统存在的错误与不足. 经过形式化建模与验证后, 可在系统实现前发现潜在问题, 避免了后期维修与调试成本. 形式化方法也有助于开发人员明确系统逻辑关系、消除歧义与模糊、保证系统按期望方式工作. 另一重要优点在于形式化方法能给出高可信度的论证与保障. 利用定理证明和形式推理技术, 可以严格证明系统的正确性和满足特定的性质 (如安全性、活性等). 形式化方法的证明过程是建立在数学逻辑与推理规则的基础之上的, 是严密可靠的. 这一严谨的论证能够为该系统设计与实施提供自信与保证, 有力地支撑系统的可靠性. 尽管形式化方法在实践中可能面临如复杂性、学习曲线和可扩展性等挑战, 但它仍然是一个强大而有效的工具, 对于开发和验证关键系统具有重要意义. 形式化方法已经在如航空航天, 铁路信号系统, 密码学以及安全协议中得到了广泛的应用, 从而保证系统正确安全.

3 基于形式化方法的区块链漏洞检测模型构建

本文提出了一种基于形式化方法的区块链系统漏洞检测模型构建方法. 首先, 分析了区块链系统中的漏洞, 并利用形式化理论定义了区块链系统的漏洞点和安全规约. 然后, 设计了一种基于 BPEL 的模型构建算法, 结合形式化方法来构建区块链漏洞检测模型. 最后, 使用模型检查器验证了所提出的模型在漏洞检测中的有效性.

3.1 区块链系统威胁分析

在对区块链系统进行漏洞检测模型构建时, 应该从区块链系统的技术架构层面出发, 依次分析数据层、网络

© 中国科学院软件研究所 http://www.jos.org.cn

层、智能合约层、共识层以及应用层面临的威胁.

3.1.1 数据层威胁分析

在区块链数据层, 数据的安全性和完整性是至关重要的. 区块链使用加密算法和哈希函数来确保数据的不可篡改性, 同时使用访问控制和隐私保护措施帮助保护敏感数据, 确保只有授权的参与者可以访问和修改数据. 然而, 区块链数据层仍然面临量子攻击、密钥管理不当、交易关联性紧密和密码组件代码漏洞等安全性威胁.

例如, 交易延展性攻击[37]就是一种针对数据层代码漏洞实施的攻击, 常被用于针对比特币交易平台的攻击. 攻击者首先向交易平台提交取款申请, 交易平台创建一笔交易支付给攻击者一笔比特币. 当监听到这笔交易时, 攻击者利用字符串填充或者其他编码方式对这笔交易的签名部分进行编码, 但不破坏签名本身. 然后, 攻击者根据更改后的交易重新生成 TXID 标识符来伪造一笔新的交易, 将伪造的交易广播到网络中. 网络中的矿工会有一定概率率先将伪造交易写入区块链, 使得原有效交易被判为双重支付[38], 导致交易平台认为原交易并未被矿工验证通过, 不得不产生一笔新交易再一次支付给攻击者. 攻击一旦成功, 攻击者就会获得双倍的比特币.

3.1.2 网络层威胁分析

区块链网络层面临的安全威胁主要有 P2P 安全漏洞、网络拓扑被用于攻击以及网络层面的隐私问题等.

P2P 网络为对等网络环境中的节点提供一种分布式、自组织的连接模式, 缺少身份认证、数据验证、网络安全管理等机制. 攻击者可以自由发布非法内容, 传播蠕虫、木马、病毒, 甚至实施分布式拒绝服务攻击 (distributed denial of service, DDoS)[39]、路由攻击[40]等, 具有不易检测、传播迅速等特点. 由于 P2P 网络采用不同于 C/S 网络的对等工作模式, 无法使用防火墙、入侵检测等技术进行有针对性的防护, 网络中的节点更易遭受攻击. 另外, P2P 网络中节点也不是完全平等的, 节点的权限会因加入网络的先后顺序而有所差异. 越先加入网络的节点占据的资源越多, 越可能限制新加入节点享有数据资源和操作权限. 因此, 在 P2P 网络上建立的区块链也会存在各节点享有资源和权限不均等的情况, 轻节点容易受到全节点的限制.

节点的网络拓扑结构会为攻击者寻找攻击目标并实施攻击创造便利. 日蚀攻击 (eclipse attack)[41]就是攻击者利用节点间的拓扑关系实现网络隔离的一种典型攻击方式. 其基本思想是攻击者通过网络拓扑控制目标节点的数据传入传出节点, 限制目标节点与外界的数据交互, 甚至将目标节点与区块链主网络隔离, 使目标节点仅能接收到攻击者传输的消息, 导致目标节点保存的区块链视图与主网区块链视图不一致, 破坏局部的一致性.

3.1.3 共识层威胁分析

区块链上的共识机制发展尚不完善, 普遍存在安全性证明不完备、安全性假设不可靠、扩展性差、一致性不稳定、初始化和重构难等问题[42]. 不同类型的共识机制还面临不同的攻击威胁. 其中, 典型的 PoW 工作量证明机制面临的主要安全威胁是双花攻击 (double spending attack)[43], 双花攻击是指攻击者企图在区块链上记录一笔与现有区块链上的交易相违背的无效交易. 常用的攻击方法是产生一条更长的区块链分叉, 使包含原交易的区块链被大多数旷工丢弃. BFT 拜占庭容错共识机制面临的主要安全威胁是女巫攻击, 女巫攻击 (sybil attack)[44]是一种在点对点 (P2P) 网络中的攻击形式: 攻击者使用单个节点在 P2P 网络中伪造多个身份, 以削弱网络的冗余, 降低网络的稳健性, 并监控或破坏正常的网络活动.

3.1.4 智能合约层威胁分析

智能合约层是区块链系统的一个重要组成部分, 它提供了在区块链上执行的自动化合约功能. 智能合约层由智能合约编程语言和相应的虚拟机或执行环境所组成. 智能合约代码开源、设计数字资产转移的特点, 使得智能合约层面临代码漏洞、外部数据源调用、缺乏形式化验证、难以实现隐私保护等[45]问题. 常见的智能合约代码漏洞包括交易依赖攻击、时间戳攻击、可重入漏洞、整数溢出漏洞等[46,47].

3.1.5 应用层威胁分析

应用层面临的威胁主要由非法用户接入、弱口令、数据非授权访问、监管缺失以及用户隐私窃取等产生[48]. 由于区块链采用独特的信息传递机制, 每个轻节点只保存有限的数据, 想要查询某个交易数据时, 首先查看自身节点保存的数据, 如果未找到, 将请求相邻节点层层传递直到获取目标信息, 每个节点都可以获取到区块链上的全部

© 中国科学院软件研究所 http://www.jos.org.cn

信息, 这可能导致用户隐私数据的泄露.

区块链系统中用户主要使用虚拟地址来进行数据交互, 但是随着数据挖掘技术的发展, 通过分析现实世界和虚拟世界用户数据的种种联系, 有可能计算出一些虚拟账户的真实地址并且和相关交易信息关联起来, 从而达到盗取某些信息的目的.

3.2 区块链系统漏洞检测模型的形式化定义

由于区块链系统本质上是一个去中心化的分布式系统, 而智能合约是运行在区块链上以支持创建去信任协议的应用程序或程序, 因此需要考虑智能合约层特有的漏洞. 同样重要的是要考虑一般分布式系统中存在的攻击者可以发起攻击行为的主客观条件, 包括节点类型、系统状态、安全属性以及节点之间的信任关系[49]. 要对区块链系统进行漏洞检测, 需要综合考虑多种影响区块链系统安全的因素, 因此本文给出结合多重安全因素的漏洞检测模型的形式化定义.

定义 1. 区块链系统的漏洞检测模型 VDMBS = (F, H, R, I, S, A, G), 其中, F 是区块链系统中服务或功能模块的集合, H 是在区块链系统中主机的集合, R 是主机之间的连接关系集合, I 表示入侵者的许可, S 表示区块链系统的状态集合, A 表示改变漏洞状态的行为集合, G 表示区块链系统运行过程中应满足的一组安全属性. 它们的具体定义如下.

(1) 区块链系统中服务或功能模块 F 的集合, 例如发送函数 Transfer(from, to, amount) $\in F$. 区块链系统由一组服务或功能组成, 其中功能点 (表示为 f) 是进行区块链安全测试的基本单元.

(2) $H = (ID_H, User, Svcs, Sw, Vuls)$, 其中 ID_H 是主机的唯一标识符, $User$ 是使用主机的用户类型, $Svcs$ 表示主机上运行的服务, Sw 是安装在主机上的软件, $Vuls$ 是主机上的漏洞集合.

(3) 主机之间的交互关系 R 可以描述为三元关系 $R \subseteq Host \times Host \times A$. 例如, $R(h_1, h_2, a)$ 表示主机 h_1 可以通过行为 a 与主机 h_2 交互.

(4) $I = (P_i(h), A_i)$, 其中 $P_i(h) \rightarrow \{None, User, Root\}$ 表示入侵者 i 在主机 h 上的权限级别, 权限级别满足全序关系: $None < User < Root$. A_i 描述了入侵者 i 执行的行为.

(5) $S = (ID_S, Name, Sub, Obj, Env)$, 其中 ID_S 是状态的编号, $Name$ 是状态名称, Sub 是主体条件, Obj 是客体条件, Env 是环境条件, 是执行下一步操作的前提条件.

(6) $A = (ID_A, Name, S_s, S_d, \lambda)$, 其中 ID_A 是行为的编号, $Name$ 是行为的名称, S_s 是该行为的源状态, S_d 是该行为的目标状态行为, λ 是衡量该行为难易程度的成本参数. 行为的执行将使漏洞状态图移动到下一个状态, 下一个状态可能是执行下一个行为的前提.

(7) $G = (ID_G, Des, Ctl)$ 表示区块链系统运行过程中需要满足的安全保护目标, 其中 ID_G 是安全属性的编号, Des 是该安全属性的详细描述, Ctl 是该安全属性用分支时序逻辑进行描述的形式化规约. 例如, 如果区块链系统的安全属性是主机 h_1 不信任主机 h_2, 那么在区块链系统运行过程中, 主机 h_2 无法访问 h_1, 需要保证 $AG(!Rsh\text{-}Trust(h_1, h_2))$.

3.3 基于 BPEL 的区块链系统漏洞检测模型

针对区块链系统的特点, 本文提出了一种基于业务流程执行语言 (business process execution language, BPEL) 的区块链系统漏洞检测模型 VDMBS. 首先, 本文通过分析区块链系统中不同类型的用户主机以及运行在其上面的服务或功能来构造主机集合. 接着, 本文应用解耦的方法把整个系统从每个操作对象模块中分离出来成为一组系统模块的集合. 本文还基于 BPEL 流程[50]循环构建每个模块的模型, 并描述区块链系统每个独立模块的状态迁移、行为和环境条件, 完成功能、状态和行为的模型. 在循环结束后, 建立交互模块模型与节点模型以反映系统模块的状态以及相互关联的模块和节点之间的交互事件, 从而完成区块链系统的信任关系模型构建. 然后, 在网络攻击分析的基础上构建入侵者模型, 模拟入侵者对区块链系统进行网络攻击的行为, 并构建相应的状态机图. 最后, 基于构建的信任关系和客观条件, 提供了安全属性的形式化规约定义, 确保区块链系统的正确运行. 区块链系统漏洞检测模型构建流程如图 1 所示, 构建算法如算法 1 所示.

© 中国科学院软件研究所 http://www.jos.org.cn

算法 1. 基于 BPEL 的区块链系统漏洞检测模型构建算法.

输入: 区块链系统;

输出: 区块链系统漏洞检测模型 (VDMBS).

1. Construct *system* = Process(*Blockchain System*) //对区块链系统进行预处理
2. *functionSet* = null, *hostSet* = null, *trustrelationshipSet* = null, *intruderSet* = null, *stateSet* = null, *actionSet* = null, *goatSet* = null //定义区块链系统运行期间的功能、主机节点、信任关系、入侵者、状态、行为和安全目标的集合
3. VDMBS = (*functionSet*, *hostSet*, *trustrelationshipSet*, *intruderSet*, *stateSet*, *actionSet*, *goatSet*} //定义区块链系统漏洞模型的数据结构
4. *functionSet* = moduleDivision(functionalAnalysis()) //分析功能并执行应用解耦
5. **for each** *functionSet* **do**
6. *StateMachineDiagram* = createStateMachineDiagram (*system*) //建立当前模块的状态机图
7. execitAnalysis(*StateMachineDiagram*) //分析状态机图
8. Return *stateSe*t, *actionSet*, *hostSet* //完成当前功能模块的状态、主机和动作的构建
9. **end for**
10. *InteractionModel* = createInteractionModel(*functionSet*, *stateSet*, *actionSet*, *hostSet*) //构建交互模型
11. *trustrelationshipSet* = createTrustRelationship(*InteractionModel*) //构建信任关系模型
12. *intruderset* = createIntruder(*intruder*) //建立入侵者模型
13. *goatSet* = createGoatSet(*statement of requirements*) //构建安全属性规约
14. Return VDMBS

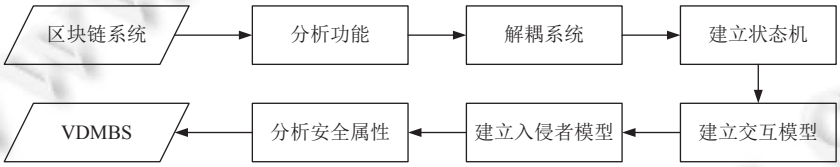

图 1 区块链系统漏洞检测模型构建流程

3.3.1 功能模块、主机、状态和行为的构建

区块链系统可以通过发送和接收消息来改变主机的内部状态. 主机从其他主机接收输入消息, 然后对这些消息进行处理并发送到其他主机, 并且该主机从一种状态迁移到另一种状态. 区块链系统本质上可以视为是一个状态迁移系统.

有限状态自动机是描述有限状态迁移系统的抽象数学模型, 而 NuSMV 模型是在有限状态自动机的基础上构造的, 因此用它来对区块链系统进行建模是非常合适的. 首先, 本文通过对区块链系统中不同类型的用户主机以及主机上运行的服务或功能进行分析来构造主机集合. 接着, 本文提出了一个基于消息会话的区块链有限状态自动机模型, 并且以接收和发送消息作为状态迁移的条件, 构建区块链系统漏洞检测模型的状态和行为.

定义 2. 区块链系统的有限状态自动机模型 $FSAMBS=(S, S_O, S_F, M_S, M_R, \delta)$ 用六元组表示, 其中 S 是有限状态集, S_O 是初始状态, S_F 是终止状态的集合, M_S 是发送消息集的集合, M_R 是接收消息集的集合, δ 表示状态迁移函数, 可以表示为 δ: $S\subseteq M_S\rightarrow S$ 或 δ: $S\subseteq M_R\rightarrow S$, $\delta(s, m_S)=s'$表示 *FSAMBS* 在状态 s 发送消息后将状态迁移到状态 s', 同时 $\delta(s, m_R)=s'$ 表示 *FSAMBS* 在状态 s 接收消息后将状态迁移到状态 s'.

如图 2 所示的状态迁移图是一个具有节点和有向弧的有向图, 它是表示并发程序的图形建模方法. 如图 2 所示, 若有限状态自动机满足条件 a, 则从源状态迁移到目标状态.

© 中国科学院软件研究所 http://www.jos.org.cn

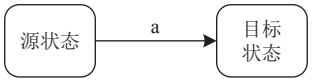

图 2 状态迁移图

BPEL 流程由活动组成, BPEL 活动又由基本活动和结构化活动组成. 以下是基于消息会话的 FSAMBS 的主要基本活动、结构化活动以及转换过程, 其中基本活动是构建块的最小单元, 且转换后不能扩展.

(1) 基本活动的转换

1) <receive> 活动

<receive> 活动用于接收来自功能服务的调用请求, 当其执行完成时, 服务本身的状态将被迁移. 转换过程如图 3 所示.

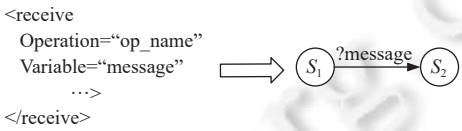

图 3 <receive> 活动转换为有限状态机图

2) <reply> 活动

当收到<receive> 活动时, <reply> 活动响应功能服务; 当它的执行完成后, 其状态会发生迁移. 具体的转换过程如图 4 所示.

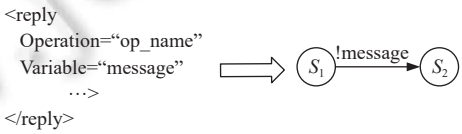

图 4 <reply> 活动转换为有限状态机图

3) <invoke> 活动

<receive> 活动用于调用功能服务, 它首先向功能服务发送消息以请求调用, 然后从功能服务接收响应消息. 具体转换过程如图 5 所示.

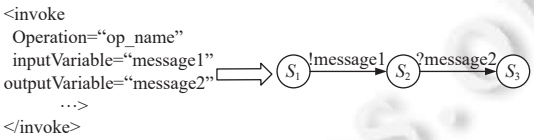

图 5 <invoke> 活动转换为有限状态机图

(2) 结构化活动的转换

1) <sequence> 活动

<sequence> 活动包含<sequence> 元素中出现的一个或多个按照词汇顺序执行的活动. 当序列中的最后一项活动完成时, <sequence> 活动完成. 具体的转换过程如图 6 所示.

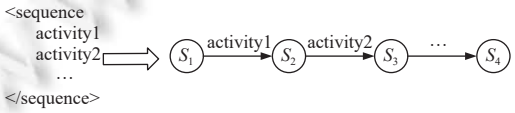

图 6 <sequence> 活动转换为有限状态机图

© 中国科学院软件研究所 http://www.jos.org.cn

2) <switch> 活动

<switch> 活动包含两个或多个以 XPath 表达式形式表示的分支. 如果表达式为 true, 则执行该分支; 否则, BPEL 流程将移至下一个分支条件, 直到找到有效的分支条件、遇到其他分支或执行完分支. 如果多个分支条件为 true, 则 BPEL 执行第 1 个 true 分支. 具体转换流程如图 7 所示.

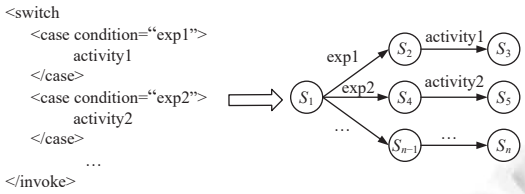

图 7 <switch> 活动转换为有限状态机图

3) <while> 活动

<while> 活动用于重复执行一个活动或一组活动, 直到满足指定的条件为止. 具体转换流程如图 8 所示.

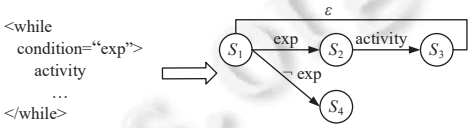

图 8 <while> 活动转换为有限状态机图

4) <repeatUntil> 活动

<repeatUntil> 活动重复执行一个活动或一组活动, 直到满足指定的条件为止, 每次执行循环体后都会测试条件. 与<while> 活动相反, <repeatUntil> 循环至少执行一次所包含的活动. 具体转换流程如图 9 所示.

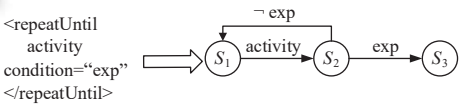

图 9 <repeatUntil> 活动转换为有限状态机图

5) <flow> 活动

<flow> 活动用于并行执行多个活动, 可以同时处理多个任务. 当<flow> 包含的所有活动都已完成时, <flow> 完成. 具体转换流程如图 10 所示.

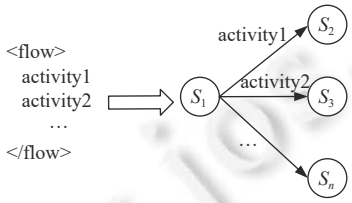

图 10 <flow> 活动转换为有限状态机图

3.3.2 主机及交互关系的构建

区块链系统中的交互关系是指区块链系统中每个独立主机之间的交互关系. 一个良好的信任关系不仅可以反映出个体之间正确的交互, 又能够有利于构建主机之间交互关系的安全属性规约. 然后, 用 *FSAMBS* 对主机的状态变化、条件因素和触发事件进行描述, 从而完成区块链系统漏洞检测模型的状态和行为的构建. 然而, 目前对于区块链系统中所有主机之间的交互关系还缺乏全局描述. 为了更清晰地描述主机之间的交互关系, 本文在上述有限状态自动机模型的基础上对区块链系统的交互关系进行建模.

© 中国科学院软件研究所 http://www.jos.org.cn

定义 3. 区块链系统交互模型 $IMBS = FSAMBS_1 \| FSAMBS_2 \| \ldots \| FSAMBS_n = (S_{1\|2\|\ldots\|n}, S_{O_{1\|2\|\cdots\|n}}, S_{F_{1\|2\|\cdots\|n}}, M_{S_{1\|2\|\cdots\|n}}, M_{R_{1\|2\|\cdots\|n}}, \delta_{1\|2\|\ldots\|n})$. $FSAMBS_n$ 是区块链系统的第 n 个有限状态自动机模型, 它们的交互模型被建模为具有 n 个提供服务主机的同步自动机. $S_{1\|2\|\ldots\|n} = S_1 \times S_2 \times \ldots \times S_n$ 是同步自动机的状态集, $S_{O_{1\|2\|\cdots\|n}} = \{S_{O_1}, S_{O_2}, \ldots, S_{O_n}\}$ 是同步自动机的初始状态, $S_{F_{1\|2\|\cdots\|n}} = S_{F_1} \times S_{F_2} \times \ldots \times S_{F_n}$ 是同步自动机的终止状态, $M_{S_{1\|2\|\cdots\|n}} = M_{S_1} \cup M_{S_2} \cup \ldots \cup M_{S_n}$ 是同步自动机, $M_{R_{1\|2\|\cdots\|n}} = M_{R_1} \cup M_{R_2} \cup \ldots \cup M_{R_n}$ 是同步自动机的接收消息, $\delta_{1\|2\|\ldots\|n}$: $S_{1\|2\|\ldots\|n} \subseteq M_{S_{1\|2\|\cdots\|n}} \rightarrow S_{1\|2\|\ldots\|n}$ or δ: $S_{1\|2\|\ldots\|n} \subseteq M_{R_{1\|2\|\cdots\|n}} \rightarrow S_{1\|2\|\ldots\|n}$ 是同步自动机的状态迁移函数.

3.3.3 入侵者模型的构建

在对区块链系统运行过程中节点主机的交互和安全属性进行建模和分析的同时, 还需对入侵者的能力进行分析. 本文将攻击者模型定义为 $I = (P_i(h), A_i)$, 其中 $P_i(h) \rightarrow \{None, User, Root\}$ 表示入侵者 i 对主机 h 的权限级别, 权限级别满足全序关系: $None < User < Root$. A_i 描述入侵者 i 执行的行为, 在区块链系统运行的真实环境中入侵者类似于系统的参与者, 可以作为响应方参与到系统的执行中. 除了系统参与者的行为外, 入侵者还可以在区块链系统的运行过程中对系统造成攻击. 入侵者模型对入侵者的行为描述如下.

(1) 入侵者可以向系统发送任何新信息.

(2) 入侵者可以调用消息来产生不同的攻击路径.

(3) 入侵者可以拥有密钥并使用它来实现攻击模型.

(4) 入侵者可以成为合法用户进行操作, 并可以在公共通道上与其他用户通信.

(5) 入侵者可以构建恶意代码, 并在代码被触发时实现恶意攻击.

在建模过程中, 入侵者模型允许通过放置在公共频道的 $Out(k)$ 有选择地将指定的信息 K 泄露给攻击者, 入侵者利用信息 K 重复以下操作: 从公共频道获取信息 K, 调用信息 K 生成不同的攻击路径和方法, 分析假设诚实场景中每个参与者的潜在攻击, 如图 11 所示. 入侵者利用其生成攻击路径的能力来判断分析过程中是否存在相应的攻击漏洞, 以及攻击是否能够成功.

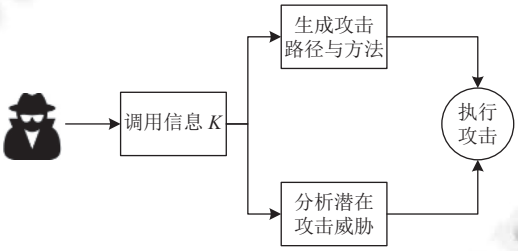

图 11 入侵者攻击生成流程

3.3.4 安全属性规约的构建

区块链系统安全属性规约是一种定义系统或软件安全属性的形式化规约, 用于确保区块链系统在设计和实施过程中满足安全要求. 下面是构建区块链系统安全属性规约的一般方法.

(1) 确定区块链系统的安全目标: 明确区块链系统所需的安全目标, 例如保护交易数据的机密性和完整性、共识协议的一致性、交易和身份的隐私保护以及智能合约安全等. 这些目标应该与特定的区块链应用场景和使用案例相匹配.

(2) 分析区块链系统的威胁: 识别和分析可能影响区块链系统安全性的威胁和攻击类型. 考虑恶意节点、51% 攻击、共识算法攻击、智能合约漏洞等常见的威胁.

(3) 确定安全属性: 根据安全目标和威胁模型, 确定区块链系统需要满足的安全属性. 这些属性可能包括数据保密性、数据完整性、节点身份验证、抗攻击性能等.

(4) 定义安全规则和机制: 为每个安全属性定义具体的规则、机制和要求. 例如, 规定只有经过身份验证的节点才能参与共识过程, 采用加密算法保护数据的机密性, 设计安全审计机制等.

© 中国科学院软件研究所 http://www.jos.org.cn

(5) 使用形式化语言描述规约: 选择适当的形式化语言来描述区块链系统的安全属性规约, 本文采用线性时序逻辑 LTL 和分支时序逻辑 CTL.

3.4 基于 NuSMV 的形式化验证

区块链系统复杂性高、边界模糊等问题给区块链系统的安全测试带来了很大的困难. 因此, 在测试阶段有必要对区块链系统进行解耦, 将功能模块分离成单元进行功能测试, 同时将其整合到系统测试的结果中. 在本文中, 将独立运行的对象模块分离出来, 得出一组系统模块, 并构建每个模块的模型. 最后, 本文对各个模块构建的安全属性规约进行验证, 检查其是否满足系统的安全要求.

在模型检查的过程中, 区块链系统需要一个形式化的验证工具 NuSMV[51]来测试其安全性. NuSMV 是一个结构良好、开放、灵活、有记录的符号模型检查器, 可以用来检查预定义的安全属性形式化规约是否适用于定义的模型. 因此, 本文采用 NuSMV 作为形式验证工具来检查安全属性规约是否符合区块链系统的安全要求. 如果模型检查器输出为真, 则不存在不安全的状态; 相反, 它违反了安全规约, 表明区块链系统有潜在的脆弱性.

4 实例分析

本节以一个简单的区块链系统——基于区块链的电子投票系统[52]为例, 说明区块链系统漏洞检测模型的可行性和有效性. 这个区块链系统由 5 个过程组成, 包括: 选举创建、选民登记、投票交易、统计结果和验证投票.

4.1 基于区块链的电子投票系统

基于区块链的电子投票系统由 4 个功能组成, 分别是投票人、管理员、智能合约和验证功能. 各功能之间的交互如下.

(1) 投票人向管理员发送认证请求信息“*credentials*”.

(2) 管理员收到“*credentials*”消息后, 若返回“*wallet_id*”消息则认证成功.

(3) 投票人向管理员发送查询请求消息“*request*”.

(4) 管理员收到“*request*”信息后向智能合约模块发送“*candidates_req*”信息; 如果智能合约模块返回“*candidates_list*”信息, 则候选人信息被成功搜索到.

(5) 此时, 管理员向投票人发送“*candidates_info*”的消息, 否则向投票人发送“*no*”的消息.

(6) 在收到“*candidates_info*”信息后, 投票人向智能合约模块发送投票请求信息“*sign_vote*”.

(7) 在收到“*sign_vote*”消息后, 智能合约模块发送验证请求消息“*vote_verify*”, 模块中的节点主机对投票进行验证, 如果验证成功则返回消息“*transaction_id*”.

4.2 区块链系统的漏洞检测模型

基于第 3.2 节提出的区块链系统漏洞检测模型的形式化定义和第 3.3 节提出的模型构建方法, 本文构建了面向区块链电子投票系统的漏洞检测模型.

第 1 步: 分析区块链系统的功能模块, 它们的描述如表 3 所示.

表 3 区块链电子投票选举系统功能模块

功能模块	功能模块描述
投票人模块	为用户提供投票登记、投票等服务
管理员模块	验证选民身份并创建选举
智能合约模块	提供区块链和用户之间的交互
验证模块	验证与投票相关的信息, 并将成功验证的信息添加到区块链中
区块链模块	为区块链系统提供数据库服务

根据分析可知区块链系统的功能模块有投票人模块、管理员模块、智能合约模块、验证模块、区块链模块, 将这 5 个模块定义为功能模块集 F = {Voter, Admin, SCM, VM, BC}.

© 中国科学院软件研究所 http://www.jos.org.cn

第 2 步: 在实验系统的区块链网络中, 不同的主机为 5 个功能模块服务. 根据所使用的区块链系统的特点, 需要建模的主机是管理员的子集、选民的子集、地区节点主机的子集和引导节点主机的子集, 它们被标记为 *Admin_h*、*Voter_h*、*Districtnode_h*、*Bootnode_h*. 得到的主机集如表 4 所示.

表 4 区块链电子投票选举系统主机集

主机子集	编号 (ID_H)	用户 (*User*)	服务 (*Svcs*)	安装软件 (*Sw*)	漏洞 (*Vuls*)
Admin_h	*ah*	Administrator	Admin	software agent	none
Voter_h	vh_i	Voter	Voter	software agent	none
Districtnode_h	dh_i	Verifier	VM	software agent	none
Bootnode_h	dh_i	Agency	SCM	software agent	none

第 3 步: 根据区块链系统的运行情况, 先将系统运行的各个模块的交互用 BPEL 语言描述. 图 12 展示部分投票过程的 BPEL 描述.

```
<bpel:invoke partnerLink="Voter" operation="send_sign_vote">
    <bpel:input variable="voterAddress" />
    <bpel:input variable="candidatesIndex" />
</bpel:invoke>
```

图 12 投票过程的 BPEL 描述

接着, 根据 BPEL 语言描述的业务流程和模块间的交互构建各个模块的有限状态机. 描述在投票主机上提供的投票服务的有限状态机可以表示为 $S=\{v_0, v_1, v_2, v_3, v_4, v_5, v_6\}$, 其中初始状态 $S_O=v_0$, 终止状态的集合 $S_F=\{v_6\}$, 发送消息的集合为 $M_S=\{credentials, request, sign\_vote\}$, 接收消息的集合为 $M_R=\{wallet\_id, candidates\_info, transaction\_id, no, candidates\_no, transaction\_no\}$, 状态迁移为 $\delta(v_0, credentials)=v_1$, $\delta(v_1, wallet\_id)=v_2$, $\delta(v_1, no)=v_6$, $\delta(v_2, request)=v_3$, $\delta(v_3, candidates\_info)=v_4$, $\delta(v_4, sign\_vote)=v_5$, $\delta(v_4, candidates\_no)=v_6$, $\delta(v_5, transaction\_id \mid transaction\_no)=v_6$, 投票人模块有限状态机图如图 13 所示. 同样, 管理员模块如图 14 所示. 智能合约模块如图 15 所示. 验证模块如图 16 所示. 区块链模块如图 17 所示.

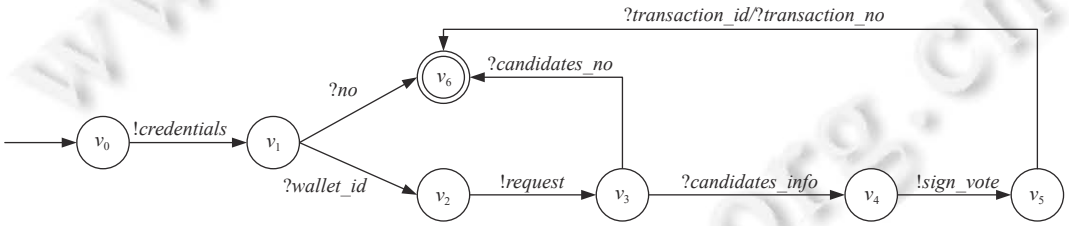

图 13 投票人模块有限状态机图

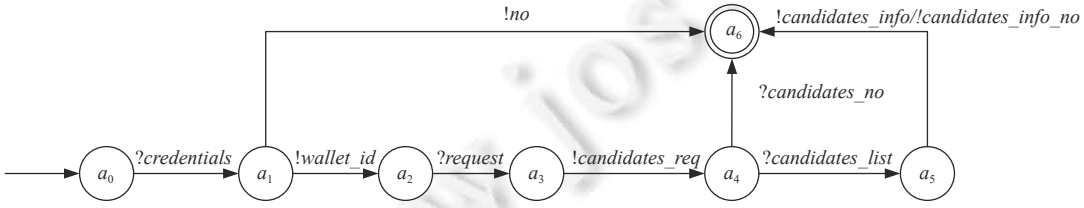

图 14 管理员模块有限状态机图

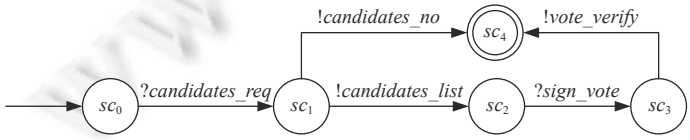

图 15 智能合约模块有限状态机图

© 中国科学院软件研究所 http://www.jos.org.cn

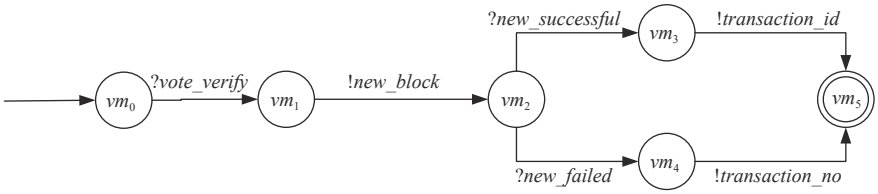

图 16　验证模块有限状态机图

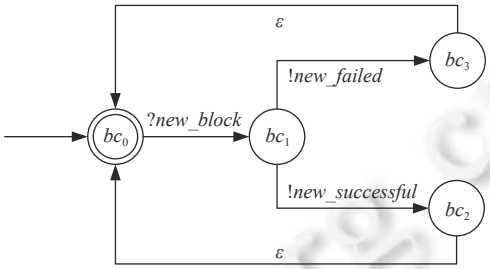

图 17　区块链模块有限状态机图

第 4 步: 在以上 5 个功能模块的有限状态机基础上, 构建区块链系统的交互模型为 $IMBS = FSAMBS_{\text{Voter}} \| FSAMBS_{\text{Admin}} \| FSAMBS_{\text{SCM}} \| FSAMBS_{\text{VM}} \| FSAMBS_{\text{BC}} = (S_{\text{Voter}\|\|\text{Admin}\|\|\text{SCM}\|\|\text{VM}\|\|\text{BC}}, S_{O\text{Voter}\|\|\text{Admin}\|\|\text{SCM}\|\|\text{VM}\|\|\text{BC}}, S_{F\text{Voter}\|\|\text{Admin}\|\|\text{SCM}\|\|\text{VM}\|\|\text{BC}}, M_{S\text{Voter}\|\|\text{Admin}\|\|\text{SCM}\|\|\text{VM}\|\|\text{BC}}, M_{R\text{Voter}\|\|\text{Admin}\|\|\text{SCM}\|\|\text{VM}\|\|\text{BC}}, \delta_{\text{Voter}\|\|\text{Admin}\|\|\text{SCM}\|\|\text{VM}\|\|\text{BC}})$. 系统的交互过程如图 18 所示.

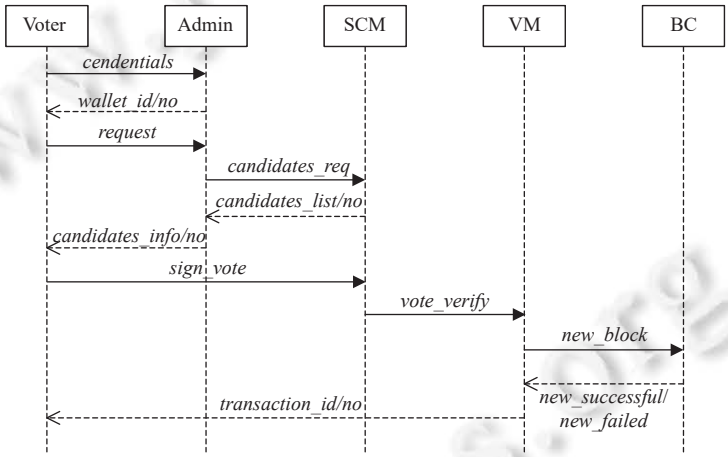

图 18　区块链电子投票系统交互过程图

第 5 步: 为了对区块链系统进行形式化的分析, 对入侵者的行为进行形式化分析, 从而完成潜在漏洞的形式化建模与分析.

假设外部入侵者必须控制 n 台主机才能对区块链系统进行有效的分布式拒绝服务攻击, 入侵者必须首先控制 $h_1, h_2, \ldots, h_n$ 的主机. 如果主机中存在程序缓冲区溢出, 入侵者就会利用主机的缓冲区溢出来获得被破坏主机的 root 权限. 因此, 为了对区块链系统进行有效的拒绝服务攻击, 入侵者采取了以下攻击过程. 它首先利用缓冲区溢出获得 n 台主机, 并在实验系统网络中的 n 台主机上获得管理员权限; 然后, 它在主机上安装代理并控制其与区块链网络中的其他主机进行交互, 并向目标主机或区块链发送大量数据包, 从而使目标主机发生拒绝服务.

基于给定的漏洞建模方法, 表 5 详细说明了入侵者进行区块链拒绝服务攻击的行为, 入侵者的有限状态机图如图 19 所示.

© 中国科学院软件研究所 http://www.jos.org.cn

表 5 区块链拒绝服务攻击行为集

行为编号	行为名称	源状态	目标状态
ac_1	缓存溢出攻击	$P_i(h_i) \geqslant User$	$P_i(h_i) = Root$
ac_2	在h_i上安装代理	$RP_i(h_i) = Root$	agent_{h_i}
ac_3	控制h_i向目标主机发送大量数据包	$P_i(h_i) = Root$	目标主机拒绝服务

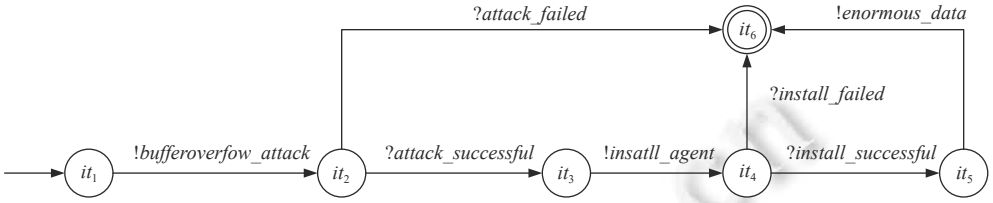

图 19 区块链拒绝服务攻击状态机图

第 6 步: 由于服务与许多行为交互, 而行为只有两种状态 (包括发送消息和接收消息), 因此用数组变量来表示行为, 每个数组元素对应一个行为, 如表 6 所示.

表 6 消息数组

行为编号	行为名称	行为编号	行为名称
message[1]	*cendentials*	*message*[9]	*candidates_info_no*
message[2]	*wallet_id*	*message*[10]	*sign_vote*
message[3]	*wallet_no*	*message*[11]	*vote_verify*
message[4]	*request*	*message*[12]	*new_block*
message[5]	*candidates_req*	*message*[13]	*new_successful*
message[6]	*candidates_list*	*message*[14]	*new_failed*
message[7]	*candidates_no*	*message*[15]	*transaction_id*
message[8]	*candidates_info*	*message*[16]	*transaction_no*

第 7 步: 前面的步骤构建了状态迁移图和区块链系统不同主机上的功能模块的交互过程. 为了在形式上验证系统, 有必要使用逻辑来描述其属性. 首先, 综合考虑区块链系统中的加密技术、身份验证机制、网络安全等, 确定区块链系统的安全目标, 如表 7 所示.

表 7 实验系统安全目标表

安全目标	具体描述
防篡改性	确保投票信息的完整性, 防止任何人在投票过程中篡改或操纵选票. 区块链技术提供了去中心化和分布式的特性, 使得投票记录无法被单一实体篡改, 从而确保了数据的可信度
隐私保护	确保选民的隐私和匿名性不被泄露. 选民的身份和投票信息应当得到适当的加密和保护, 以防止未经授权的访问或信息泄露
确保身份验证	确保只有符合条件的选民才能参与投票, 并防止恶意行为者使用虚假身份进行投票. 区块链技术可以用于验证选民的身份, 并提供可信的身份验证机制
防止重复投票	防止选民多次投票或重复投票, 以保证选举结果的准确性. 区块链技术可以记录每个选民的投票记录, 并确保每个选民只能投票一次
抗拒否认性	确保投票参与者无法否认自己的参与或投票行为. 区块链技术的不可篡改性和可追溯性特点可以提供不可否认性, 防止选民或其他参与者否认其投票行为
抗攻击性	系统应具备抵御各种恶意攻击的能力, 包括分布式拒绝服务攻击、恶意软件攻击、网络攻击等. 区块链技术的去中心化和分布式特性使得系统更加抗攻击, 并能够保持稳定运行
透明和可验证性	确保选民和其他参与者可以验证选举过程的公正性和透明度, 包括投票结果的计算和统计. 区块链技术提供了公开和可追溯的投票记录, 使得选民和其他参与者可以独立验证选举结果
交互正确性	区块链系统交互正确性是指确保参与方在进行交互时, 系统能够正确处理和执行其交互操作, 从而确保系统的一致性和正确性

© 中国科学院软件研究所 http://www.jos.org.cn

在确定区块链系统的安全目标后, 针对系统的安全防护目标, 分析该区块链系统面临的威胁, 如表 8 所示.

基于以上确定的区块链系统的安全目标和面临的威胁, 根据系统的具体设计文档说明等构建相应的安全属性, 并根据以上步骤构建的各模块有限状态机和交互过程用 LTL 语言或者 CTL 语言对安全属性进行形式化规约描述.

表 8 实验系统威胁分析表

安全目标	威胁	具体描述
防篡改性	投票信息篡改	攻击者可能试图篡改投票信息, 修改选民的投票选择或增加虚假的选票; 选民追踪和隐私泄露: 攻击者可能试图追踪选民的投票行为, 破坏选民的匿名性, 或通过不当手段获取选民的身份信息和投票偏好
隐私保护	选民追踪和隐私泄露	攻击者可能试图追踪选民的投票行为, 破坏选民的匿名性, 或通过不当手段获取选民的身份信息和投票偏好
确保身份验证	身份验证攻击	攻击者可能尝试冒充合法选民, 通过欺骗或破解身份验证机制获取投票权限
防止重复投票	重放攻击	攻击者可能通过重放已提交的合法投票记录来进行重复投票, 以增加特定选项的投票数
抗拒否认性	投票篡改否认	参与方可能否认对投票记录或结果进行的任何篡改或修改行为, 否认对投票过程的任何不当干预
抗攻击性	攻击威胁	系统面临51%攻击、DDoS攻击、女巫攻击、日蚀攻击等威胁, 以及智能合约漏洞面临的重入攻击等
透明和可验证性	虚假信息输入	攻击者可能通过创建虚假的投票记录或投票人身份, 输入错误的信息, 干扰投票结果的正确性和可验证性
交互正确性	恶意节点	恶意节点可能篡改或延迟传递消息, 修改或删除交互消息, 以干扰正常的投票过程或改变投票结果

(1) 防篡改性

● G1-1 防止投票数据的篡改: 对于任意一个已投票的选民, 在下两个状态中选民所投的候选人的票数递增.

$$AG(pr1.voted = \text{true}) \rightarrow AX(AX(pr5.candidates[pr1.vote].voteCount > pr5.candidates[pr1.vote].voteCount)).$$

● G1-2 防止投票结果的篡改: 对于任意一个最终获胜的候选人, 在下两个状态中, 候选人的票数递增.

$$AG(pr5.winningCandidates = c) \rightarrow AX(AX(pr5.candidates[c].voteCount > pr5.candidates[c].voteCount)).$$

(2) 隐私保护

● G2-1 选民身份的隐私保护: 对于任意一个已投票的选民, 在下两个状态中选民仍然保持已投票状态.

$$AG(pr1.voted = \text{true}) \rightarrow AX(AX(pr1.voted = \text{true})).$$

● G2-2 投票偏好的隐私保护: 对于任意一个已投票的选民, 在下两个状态中选民的投票选择保持不变.

$$AG(pr1.voted = \text{true}) \rightarrow AX(AX(pr1.vote = pr1.vote)).$$

● G2-3 选民身份与投票结果的隐私保护: 对于任意一个最终获胜的候选人, 在下两个状态中候选人的票数保持不变.

$$AG(pr5.winningCandidates = c) \rightarrow AX(AX(pr5.candidates[c].voteCount = pr5.candidates[c].voteCount)).$$

(3) 确保身份验证

● G3-1 正确的身份验证: 对于任意一个具有投票权的选民, 在下一个状态中选民的投票权仍然存在且不为 0.

$$AG(pr1.weight \mathrel{!}= 0) \rightarrow AX(pr1.weight \mathrel{!}= 0).$$

● G3-2 防止伪造身份: 对于任意一个具有投票权的选民, 在下一个状态中选民不会将其委托给自己, 防止伪造身份.

$$AG(pr1.weight \mathrel{!}= 0) \rightarrow AX(EX\ (pr1.delegate \mathrel{!}= pr1.address)).$$

(4) 防止重复投票

● G4-1 防止重复投票: 对于任意一个已投票的选民, 在下两个状态中选民仍然保持已投票状态, 防止重复投票.

$$AG(pr1.voted = \text{true}) \rightarrow AX(AX(pr1.voted = \text{true})).$$

● G4-2 防止委托投票重复: 对于任意一个已投票的选民 *msg.sender*, 在下两个状态中选民的委托投票对象保

© 中国科学院软件研究所 http://www.jos.org.cn

持不变, 防止委托投票重复.

$$AG(pr1.voted = \text{true}) \rightarrow AX(AX(pr1.delegate = pr1.delegate)).$$

(5) 抗拒否认性

● G5-1 选民无法否认其投票行为: 对于任意一个已投票的选民, 在下一个状态中选民无法否认其已投票行为.

$$AG(pr1.voted = \text{true}) \rightarrow AX(pr1.voted = \text{true}).$$

● G5-2 防止虚假的投票否认: 对于任意一个已投票的选民, 在下 3 个状态中选民无法否认其投票选择.

$$AG(pr1.voted = \text{true}) \rightarrow AX(AX(AX(pr1.vote = pr5.candidates))).$$

(6) 抗攻击性

● G6-1 针对 DDoS 攻击, 对于任意一个选民, 如果其发送的交互请求次数超过阈值 *threshold*, 则在下一个状态中其交互请求次数将小于阈值 *threshold*.

$$AG(pr\text{DDoS}.count(pr1) \geqslant pr\text{DDoS}.threshold) \rightarrow AX(pr\text{DDoS}.count(pr1) < pr\text{DDoS}.threshold).$$

● G6-2 针对 DDoS 攻击, 对于任意一个选民, 如果其发送的交互请求次数超过阈值 *threshold*, 则存在一条路径, 使得在该路径上其交互请求次数恢复到小于阈值 *threshold*, 从而确保系统的可用性.

$$AG(pr\text{DDoS}.count(pr1) \geqslant pr\text{DDoS}.threshold) \rightarrow AF(pr\text{DDoS}.count(pr1) < pr\text{DDoS}.threshold).$$

● G6-3 针对女巫攻击, 对于任意一个选民, 如果其创建的虚假身份数量超过阈值 *threshold*, 则在下一个状态中, 其所委托的不同身份数量将大于等于阈值 *threshold*.

$$AG(pr\text{DDoS}.count(pr1) \geqslant pr\text{DDoS}.threshold) \rightarrow AX(pr\text{DDoS}.count(pr\text{DDoS.distinct}(pr1.delegate)) \geqslant pr\text{DDoS}.threshold).$$

● G6-4 针对女巫攻击, 对于任意一个选民 *msg.sender*, 如果其创建的虚假身份数量超过阈值 *threshold*, 则在下一个状态中其所委托的不同身份数量将大于 1, 防止恶意控制多个身份.

$$AG(prSybil.count(pr1.voters) \geqslant prSybil.threshold) \rightarrow EX\ (prSybil.count(prSybil.distinct(pr1.delegate)) > 1).$$

● G6-5 针对日蚀攻击, 如果网络发生分区 (遭受日蚀攻击), 则在下一个状态中不存在选民.

$$AG(prEclipse.networkPartitioned()) \rightarrow AX(prEclipse.not\_exists(pr1)).$$

● G6-6 针对日蚀攻击, 如果网络发生分区 (遭受日蚀攻击), 则在下一个状态中存在选民委托给其他选民.

$$AG(prEclipse.networkPartitioned()) \rightarrow EX(prEclipse.exists(pr1.delegate)).$$

● G6-7 针对整数溢出漏洞, 对于任意一个候选人, 如果其票数大于 0, 则在下一个状态中候选人的票数将小于整数的最大值.

$$AG(pr5.candidates[c].voteCount > 0) \rightarrow AX(pr5.candidates[c].voteCount < \text{MAX\_INT}).$$

● G6-8 针对 51% 攻击, 如果存在超过网络总计算能力的 51% 的矿工参与挖矿, 则在下一个状态中参与挖矿的矿工数量将小于网络总计算能力的 51%.

$$AG(pr51\%.\text{count}(pr51\%.distinct(DistrictnodeS)) \geqslant pr51\%.networkHashRate \times 0.51) \rightarrow AX(pr51\%.count(pr51\%.distinct(DistrictnodeS)) < pr51\%.networkHashRate \times 0.51).$$

(7) 透明和可验证性

● G7-1 系统操作的透明性: 对于任意一个选民, 如果该选民存在, 则在下一个状态中该选民必须已经进行了投票操作, 确保系统操作的透明性.

$$AG(pr1.exists(voters)) \rightarrow AX(pr1.voted = \text{true}).$$

● G7-2 数据的可验证性: 对于任意一个最终获胜的候选人 *c*, 在下两个状态中该提案的票数必须大于 0, 确保数据的可验证性.

$$AG(pr5.winningCandidates = c) \rightarrow AX(AX(pr5.candidates[c].voteCount > 0)).$$

(8) 交互正确性

● G8-1 验证区块链系统中是否存在死锁状态, 即每个功能模块的状态是否达到终止状态, 这在 LTL 方程中表示.

$$G(F((((pr1.V\_State = v6 \& pr2.A\_State = a6) \& pr3.SC\_State = sc4) \& pr4.VM\_State = vm5) \& pr5.BC\_State = bc0)).$$

© 中国科学院软件研究所 http://www.jos.org.cn

● G8-2 在发送认证请求信息“*credentials*”后, 投票人必须收到管理员的回应, 要么认证成功, 要么认证不成功.

$$AG((message[2] \mid message[3]) \rightarrow message[1]).$$

● G8-3 在发送查询请求信息“*require*”后, 投票人必须收到管理员的回应, 要么是查询成功的信息, 要么是查询不成功的信息.

$$AG((message[8] \mid message[9]) \rightarrow message[4]).$$

● G8-4 对于候选人的查询请求“*candors_req*”, 必须有一个来自验证模块的回复, 要么查询成功“*candors_list*”, 要么查询失败“*candors_no*”.

$$AG((message[6] \mid message[7]) \rightarrow message[5]).$$

● G8-5 在发送查询请求消息“*new_block*”后, 投票者必须收到区块链的响应, 要么添加新区块成功, 要么失败.

$$AG((message[13] \mid message[14]) \rightarrow message[12]).$$

● G8-6 一旦投票信息“*sign_vote*”被发送, 必须有一个来自验证模块的回复, 要么是成功的信息“*transaction_id*”, 要么是失败的信息“*transaction_no*”.

$$AG((message[15] \mid message[16]) \rightarrow message[10]).$$

● G8-7 添加一个新区块失败, 没有发送成功信息.

$$AG(!(message[15]) \rightarrow message[14]).$$

● G8-8 验证系统的交互是否正确.

$$\begin{aligned}&AG(pr1.V\_State = v_0 \rightarrow (AX\ pr1.V\_State = v_1 \rightarrow (AX\ pr2.A\_State = a_1\\ &\rightarrow (AX\ pr2.A\_State = a_2 \rightarrow (AX\ pr1.V\_State = v_2 \rightarrow (AX\ pr1.V\_State = v_3\\ &\rightarrow (AX\ pr2.A\_State = a_3 \rightarrow (AX\ pr2.A\_State = a_4 \rightarrow (AX\ pr3.SC\_State = sc_1\\ &\rightarrow (AX\ pr3.SC\_State = sc_2 \rightarrow (AX\ pr2.A\_State = a_5 \rightarrow (AX\ pr2.A\_State = a_6\\ &\rightarrow (AX\ pr1.V\_State = v_4 \rightarrow (AX\ pr1.V\_State = v_5 \rightarrow (AX\ pr3.SC\_State = sc_3\\ &\rightarrow (AX\ pr3.SC\_State = sc_4 \rightarrow (AX\ pr4.VM\_State = vm_1 \rightarrow (AX\ pr4.VM\_State = vm_2\\ &\rightarrow (AX\ pr5.BC\_State = bc_1 \rightarrow (AX\ pr5.BC\_State = bc_2 \rightarrow (AX\ pr5.BC\_State = bc_0\\ &\rightarrow (AX\ pr4.VM\_State = vm_3 \rightarrow (AX\ pr4.VM\_State = vm_5\\ &\rightarrow (AX\ pr1.V\_State = v_6))))))))))))))))))))))).\end{aligned}$$

● G8-9 投票者只有在收到“*transaction_id*”信息时才能成功投票.

$$AG(!(pr1.Voted\text{=TRUE}\&pr1.V\_State = v_5)).$$

● G8-10 区块链系统能够抵御来自入侵者的拒绝服务攻击.

$$AG(!(enormous\_data\&pr\text{DDoS}.IT\_State = it_6)).$$

● G8-11 区块链系统可以抵御来自入侵者的日蚀攻击.

$$AG(!(pr1.Wallet\_id = \text{FALSE}\&message[10]\&pr3.SC\_State = sc_4)).$$

● G8-12 区块链系统能够抵御来自入侵者的女巫攻击.

$$AG(!prSybil.Wallet\_id\text{=FALSE}).$$

● G8-13 区块链系统能够抵御入侵者的双花攻击.

$$AG(EF(\text{forall } n\text{: } prDouble\_Node \mid prDouble(n) = prDouble(root))).$$

最后, 构建出该区块链电子投票系统的漏洞检测模型, 其形式化定义为 $VDMBS_{E\text{-}voting} = (F, H, R, I, S, A, G)$. 其中 7 个元素具体含义如下.

(1) F = {Voter, Admin, SCM, VM, BC}, 具体的功能模块描述如表 1 所示.

(2) H = {*Admin_h*, *Voter_h*, *Districtnode_h*, *Bootnode_h*}, 具体的主机唯一标识符、主机用户类型、主机上运行的服务、安装的软件、主机上的漏洞信息如表 2 所示.

(3) R = {R_1(*Voter_h*, *Admin_h*, *message*[1]), R_2(*Admin_h*, *Voter_h*, *message*[2]/*message*[3]), R_3(*Voter_h*, *Admin_h*, *message*[4]), R_4(*Admin_h*, *Bootnode_h*, *message*[5]), R_5(*Admin_h*, *Bootnode_h*, *message*[6]/ *message*[7]), R_6(*Admin_h*,

© 中国科学院软件研究所 http://www.jos.org.cn

Voter_h, *message*[8]/*message*[9]), R_7(*Voter_h*, *Bootnode_h*, *message*[10]), R_8(*Bootnode_h*, *Districtnode_h*, *message*[11]), R_9(*Districtnode_h*, *Bootnode_h*, *message*[12] / *message*[13])}.

(4) $I=\{I_{\text{DDoS}}, I_{\text{Sybil}}, I_{\text{Eclipse}}, I_{51\%}, \cdots, I_n\}$, 这里表示对实验系统发动 DDoS 攻击、女巫攻击、日蚀攻击等各种攻击者在系统中获得的权限, 具体权限信息如表 3 所示.

(5) $S=\{S_{\text{Voter}}, S_{\text{Admin}}, S_{\text{SCM}}, S_{\text{VM}}, S_{\text{BC}}\}$, 该实验系统的有限状态机包括投票人模块、管理员模块、智能合约模块、验证模块以及区块链模块, 具体的状态的编号、状态名称、主体条件、客体条件、环境条件等具体信息如图 13–图 17 所示.

(6) $A=\{message[i], i\in[1, 16]\}$, 行为表 4 中所示的消息数组中的行为, 每个行为都有相应的编号、行为名称、行为的源状态、行为的目标状态行为以及 λ 值, 例如 $A_{message[1]}=(message[1], cendentials, v_0, v_1, 0.5)$.

(7) G = {G1-1, G1-2, G2-1, G2-2, G2-3, G3-1, G3-2, G4-1, G4-2, G5-1, G5-2, G6-1, G6-2, G6-3, G6-4, G6-5, G6-6, G6-7, G6-8, G7-1, G7-2, G8-1, G8-2, G8-3, G8-4, G8-5, G8-6, G8-7, G8-8, G8-9, G8-10, G8-11, G8-12, G8-13}, 具体的安全属性描述以及形式化描述已在上文中详细地给出.

该实验系统的漏洞检测模型中 F, H, R, I, S, A 的定义构成了 5 个不同功能模块的有限状态机, G 的定义则是描述了 5 个有限状态机需要满足的形式化规约, 本文将使用 SMV 语言对有限状态机和形式化规约进行描述并输入到形式化验证工具 NuSMV 中, 进行形式化验证.

4.3 形式化验证及结果分析

第 4.2 节通过提出的模型构建方法为基于区块链的电子投票系统构建了漏洞检测模型. 本文将 5 个构建的模块的状态机图用 SMV 语言进行描述并分别输入到 NuSMV 中, 然后进行模型检查, 检查构建的安全属性是否满足系统的安全要求.

本文用 NuSMV 验证上述 34 个安全属性规约, 其中 9 个 (包括 G1-1, G3-2, G6-1, G6-2, G8-2, G8-3, G8-6, G8-8 和 G8-10) 违反了系统的安全属性. 本文对 9 个违反安全属性的安全属性规约进行分析, 分析结果如下.

G1-1 出现了后一个状态的候选人票数小于前一个状态的票数的情况, 然而 G6-7 却显示没有超过整数的最大值, 说明该系统对单一的数值有边界检查, 但是没有对数值加法操作进行安全检查. 因此需要对区块链系统中的投票智能合约进行审计, 如图 20 所示. 合约中候选人票数增加是加法操作, 但是没有使用安全的加法操作, 并且没有对加法结果进行溢出检查, 最终检测出存在加法上溢漏洞.

G3-2 能够委托投票人自己, 说明区块链系统没有正确的身份验证. 对系统的投票智能合约进行安全审计, 如图 21 所示. 其中, 委托授权功能没有身份验证修饰器, 因此该区块链系统存在访问控制不当的漏洞.

```
function vote(uint c) public {
    Voter storage sender = voters[msg.sender];
    require(sender.weight != 0, "Has no right to vote");
    require(!sender.voted, "Already voted.");
    sender.voted = true;
    sender.vote = c;
    candidates[c].voteCount += sender.weight;
}
```

图 20 投票智能合约溢出代码段

```
function delegate(address to) public {
    Voter storage sender = voters[msg.sender];
    require(!sender.voted, "You already voted.");
    require(to != msg.sender, "Self-delegation is disallowed.");
```

图 21 投票智能合约访问控制不当代码段

G6-1、G6-2 和 G8-10 表明选民在前一个状态请求交互的次数超过阈值, 而在下一个状态或者在某一条路径上存在请求交互次数不能恢复到阈值以下的问题, 而在超过系统阈值后系统状态机反例显示进入 it_6 状态, 说明该区块链系统拒绝进行服务, 无法抵抗拒绝服务攻击. 对系统存在的拒绝服务攻击风险进行分析, 通过对如图 22 代码片段所示的智能合约进行安全审计, 发现在系统投票智能合约中“*delegate*”函数的委托可能形成一个循环, 而合约中的循环检测机制无法正确处理这种情况, 从而导致选民可以拥有多次投票的权利, 向管理员发起大量投票请求.

G8-2、G8-3、G8-6 和 G8-8 都显示区块链系统中各个模块在进行交互时, 发出某一消息后并不能返回正确的消息. 通过对系统接口以及接口调用的检查发现系统存在交互逻辑错误, 即存在错误的业务逻辑实现漏洞.

© 中国科学院软件研究所 http://www.jos.org.cn

综上所述, 本文所提出的方法能够检测出基于区块链的电子投票系统中的智能合约漏洞、业务逻辑漏洞以及由漏洞引起的遭受攻击的风险, 最终结果如图 23 所示.

```
while (voters[to].delegate != address(0)) {
    to = voters[to].delegate;

    // We found a loop in the delegation, not allowed.
    require(to != msg.sender, "Found loop in delegation.");
}
```

图 22 投票智能合约存在拒绝服务攻击风险代码段

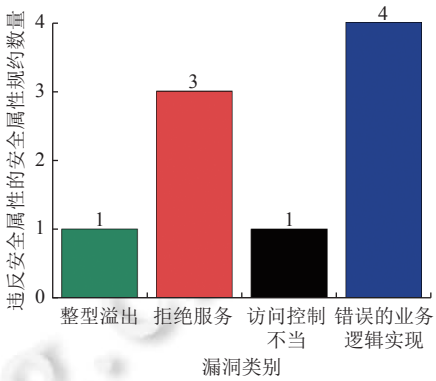

图 23 形式化验证结果图

5 实验分析

5.1 实验设置

为了验证所提出的 VDMBS 模型的有效性, 本文实现了 VDMBS 以及其他形式验证方法 (包括 F*框架[18]、ZEUS[25]) 来挖掘 5 种不同类型的区块链系统中的漏洞. 所使用的区块链系统的具体信息如表 9 所示. 验证环境如下.

表 9 实验系统介绍

系统名称	系统类型	系统来源
BES	电子投票	[52]
Polygon-NFT-marketplace	代币	[53]
AuthentiFi	身份授权	[54]
SCTS	供应链溯源	[55]
EHRs	医疗	[56]

(1) 处理器: Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz.

(2) 内存: 8 GB.

(3) 操作系统: Windows 10.

(4) 软件: Enterprise Architect.

5.2 实验结果与分析

本文对 5 种不同类型的区块链进行了漏洞建模和形式验证, 形式化验证的结果如图 24 所示. 对比实验结果如表 10 所示.

根据图 24 中所示, EHRs、PNFTM、AuthentiFi、SCTS 这 4 个区块链系统也同上文所讲解的案例系统 BES 相同构建了安全属性规约. 本文从这 4 个区块链系统的密码学、认证机制、网络安全等方面分析区块链系统的安全目标, 然后根据确定的安全保护目标分析区块链系统面临的威胁. 最后, 根据上述确定的安全目标和面临的威胁, 构建了相应的安全属性, 并用 LTL 语言或 CTL 语言对安全属性进行形式化规范描述. 其他 4 个区块链系统的安全属性也包含了案例中的安全属性, 如抗篡改、隐私保护、确保认证、抗证据否认、抗攻击、透明与认证、交互正确性等.

根据图 24 中所示 5 个区块链系统的形式化验证结果发现, 不同实验系统构建的安全属性规约数量是不同的, 经形式化验证后违反安全属性的安全属性规约数量与构建的安全属性规约的数量呈正比, 即构建的安全属性规约越多则违反安全属性的规约数量就越多.

© 中国科学院软件研究所 http://www.jos.org.cn

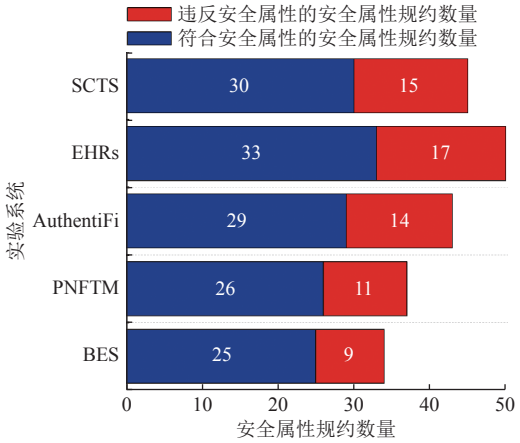

图 24 实验结果图

表 10 对比实验结果表

漏洞类型	实验系统	形式化验证工具		
		VDMBS	F*	ZEUS
错误的业务逻辑实现	BES	4	0	0
	PNFTM	4	0	0
	AuthentiFi	5	0	0
	EHRs	7	0	0
	SCTS	6	0	0
智能合约漏洞	BES	3	2	1
	PNFTM	2	1	0
	AuthentiFi	1	1	1
	EHRs	4	2	1
	SCTS	1	1	0

本文从 5 个实验系统的功能模块数量、安全属性数量以及实验系统中智能合约数量这 3 个维度对图 24 产生的现象进行了分析. 图 25 所示的是 5 个实验系统的功能模块数量、安全属性数量和智能合约数量. 从智能合约的角度出发, AuthentiFi 和 SCTs 的智能合约数量小于 PNFTM 的智能合约数量, 然而 AuthentiFi 和 SCTs 的安全属性规约数量大于 PNFTM 的安全属性规约数量, 这说明智能合约的数量对构建的安全属性规约数量影响较小. 从功能模块数量角度出发, AuthentiFi、EHRs 和 SCTs 中功能模块数量大于 BES 和 PNFTM 中功能模块数量, 而 AuthentiFi、EHRs 和 SCTs 中构建的安全属性规约都大于 BES 和 PNFTM 的安全属性规约数量, 因此实验系统中功能模块的数量对构建的安全属性规约有一定的影响. 从安全属性数量角度出发, 5 个系统的安全属性数量大小的关系为 AuthentiFi>SCTs>EHRs>PNFTM>BES, 而安全属性规约数量的大小关系也为 AuthentiFi>SCTs>EHRs>PNFTM>BES, 这说明安全属性的数量对安全属性规约的影响较大. 综上所述, 安全属性规约的数量取决于功能模块数量和安全属性数量, 而违反安全属性的规约数量则取决于构建的安全属性规约数量. 出现这种现象的原因在于安全属性规约是根据安全属性以及系统中功能模块的交互所共同构建, 即系统功能模块和安全属性越多, 说明需要验证越多的安全属性形式化规约以确保系统运行的正确性, 从而检测出系统中更多的潜在漏洞.

根据表 10 所示的对比实验结果发现所提出的 VDMBS 可以通过形式化验证手段检测到多个潜在漏洞, 其中主要分为业务逻辑实现错误漏洞和智能合约代码漏洞, 而 F*和 ZEUS 只能检测出区块链系统中的智能合约漏洞. 基于上述实验结果, 所提出的 VDMBS 在检测系统逻辑漏洞和智能合约漏洞方面优于其他区块链形式化验证工具. 为了保证实验结果的可靠性, 本文对这个实验现象进行了彻底的分析.

F*和 ZEUS 这两种形式化验证工具侧重于区块链系统中智能合约层的安全性, 而对系统运行时的整体状态关注相对较少. 然而, 所提出的 VDMBS 在设计时考虑了智能合约的交互安全以及区块链系统中其他模块的交互行为和状态迁移, 这些额外的考虑使得它能够检测到系统业务逻辑漏洞, 并且不仅限于智能合约层的漏洞检测. 与之相比, 其他两个区块链形式化验证工具在检测系统业务逻辑错误方面存在一定的限制. 在实际应用中, 区块链系统通常由多个模块和组件组成, 这些模块之间相互交互并共同协作完成系统功能. 因此, 除了智能合约层, 还需要关注其他模块之间的交互行为和状态迁移, 从而发现由于模块之间的逻辑错误和不当交互导致的系统漏洞和安全隐患. VDMBS 的提出正是为了解决这个问题. 通过使用 VDMBS, 开发者可以对区块链系统进行全面的形式化验证, 包括智能合约层和其他模块之间的交互. 这种综合的验证方法使得系统的安全性和正确性得到更全面的保证.

根据图 26 中 3 种形式化验证工具针对 5 个区块链系统漏洞检测时间的对比发现, VDMBS 消耗了更多的时间成本来提高漏洞检测的效果. 这一现象也再次证明本方法虽然在时间成本上消耗高于其他两个形式化验证工具, 但是能够更全面地对区块链系统进行检测, 检测出更多潜在的漏洞.

与其他区块链形式化验证工具相比, VDMBS 具有以下优势.

- 全面性: VDMBS 不仅关注智能合约层的漏洞, 还能检测系统逻辑漏洞和业务逻辑错误. 这使得 VDMBS 能

© 中国科学院软件研究所 http://www.jos.org.cn

够提供更全面的安全性和正确性验证.

● 综合性: VDMBS 考虑了区块链系统中各个模块之间的交互和状态迁移, 使得验证结果更贴近实际系统的运行情况.

● 可扩展性: VDMBS 综合考虑了区块链系统架构中的各种安全因素, 而现有各种大规模的区块链系统其本质上都是基于区块链架构开发而成, 因此该模型的各个组成部分基本能够覆盖各类区块链系统, 具有一定的可扩展性.

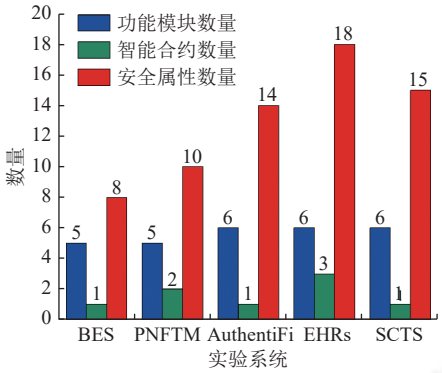

图 25 形式化验证分析图

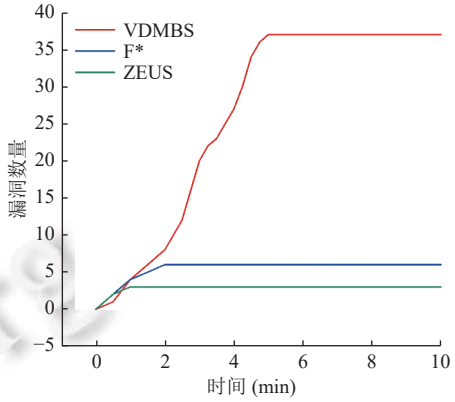

图 26 5 个区块链系统漏洞检测时间对比图

综上所述, VDMBS 在区块链系统的形式化验证方面具有独特的优势. 它能够检测到系统业务逻辑错误以及智能合约漏洞, 提供更全面的安全性和正确性验证. 在实际应用中, 开发者可以考虑采用 VDMBS 作为一种综合性的验证方法, 以保证区块链系统的可靠性和安全性.

6 总 结

随着计算机技术的快速发展, 区块链系统已逐渐成为网络应用的主要选择. 由于区块链系统的复杂性, 区块链系统存在着巨大的潜在安全风险. 然而, 各种脆弱性分析方法不能全面、统一地对区块链系统进行分析, 因而无法发现区块链系统运行过程中的各种漏洞. 针对这些问题, 考虑到影响区块链系统安全的因素, 如状态、行为、节点、功能、安全属性规约和模块交互等, 本文提出了基于形式化方法的区块链系统漏洞检测模型, 为安全分析提供理论指导. 此外, 本文还提出了一种基于 BPEL 以及形式化方法的模型构建方法, 构建更加高效、准确的区块链系统漏洞检测模型. 而后, 本文通过一个真实使用的区块链系统作为案例, 介绍了实验区块链系统的模块功能, 然后利用 NuSMV 验证了提出的漏洞模型构建算法的有效性以及漏洞检测模型的有效性. 最后, 本文对 5 种不同类型的区块链系统进行了形式化建模和验证, 以验证本文提出的方法在不同类型区块链系统上的适用性以及在检测系统交互逻辑和智能合约漏洞的能力. 此外, 还将本文提出的方法与现有的区块链系统形式化工具进行了分析和比较, 实验结果验证提出方法在检测系统逻辑漏洞和智能合约漏洞方面具有一定优势.

此外, 本文对所提的 VDMBS 模型现有的实用性和扩展性进行了详细的分析, 主要存在以下因素导致其在实用性与扩展性方面存在一定的局限性: 1) 建模复杂度高, VDMBS 模型需要构建每个功能模块的状态机, 以及节点之间的交互模型, 对于组件繁多的大型区块链系统, 建模过程较为复杂. 2) 状态空间可能爆炸问题, 在进行形式化验证时, VDMBS 模型可能导致验证工具的状态空间爆炸. 3) 需专业知识支持, VDMBS 模型的应用需要安全专家参与, 对用户的专业知识要求较高. 上述局限性在一定程度上影响了 VDMBS 模型的实用性和扩展性, 然而经过分析, VDMBS 的局限性和潜在缺点在未来可以通过继续开展相应的改进得到缓解.

在未来, 拟开发一个功能更加全面的相应的辅助建模工具, 提供状态机自动提取等功能, 缓解建模复杂度高的问题. 可以进一步关注模块划分、应用解耦和降低状态空间的复杂性, 从而解决形式化工具在验证过程中状态空间爆炸的问题. 同时, 可以优化验证算法、预集成主流区块链系统或平台、构建多场景知识库与案例库、支持增量验证等详细的方案来提高 VDMBS 模型在实际区块链系统漏洞检测中的实用性、扩展性和适用性.

© 中国科学院软件研究所 http://www.jos.org.cn

References:

[1] Latif SA, Wen FBX, Iwendi C, Wang LLF, Mohsin SM, Han ZY, Band SS. AI-empowered, blockchain and SDN integrated security architecture for IoT network of cyber physical systems. Computer Communications, 2022, 181: 274–283. [doi: 10.1016/j.comcom.2021.09.029]

[2] Kumar A, Singh AK, Ahmad I, Singh PK, Anushree, Verma PK, Alissa KA, Bajaj M, Rehman AU, Tag-Eldin E. A novel decentralized blockchain architecture for the preservation of privacy and data security against cyberattacks in healthcare. Sensors, 2022, 22(15): 5921. [doi: 10.3390/s22155921]

[3] Taylor PJ, Dargahi T, Dehghantanha A, Parizi RM, Choo KKR. A systematic literature review of blockchain cyber security. Digital Communications and Networks, 2020, 6(2): 147–156. [doi: 10.1016/j.dcan.2019.01.005]

[4] Liu ZY, Luong NC, Wang WB, Niyato D, Wang P, Liang YC, Kim DI. A survey on blockchain: A game theoretical perspective. IEEE Access, 2019, 7: 47615–47643. [doi: 10.1109/ACCESS.2019.2909924]

[5] Vasek M, Thornton M, Moore T. Empirical analysis of denial-of-service attacks in the Bitcoin ecosystem. In: Proc. of the 2014 Financial Cryptography and Data Security: FC 2014 Workshops, BITCOIN and WAHC 2014. Christ Church: Springer, 2014. 57–71. [doi: 10.1007/978-3-662-44774-1_5]

[6] Bhutta MNM, Khwaja AA, Nadeem A, Ahmad HF, Khan MK, Hanif MA, Song HB, Alshamari M, Cao Y. A survey on blockchain technology: Evolution, architecture and security. IEEE Access, 2021, 9: 61048–61073. [doi: 10.1109/ACCESS.2021.3072849]

[7] Khan MA, Salah K. IoT security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems, 2018, 82: 395–411. [doi: 10.1016/j.future.2017.11.022]

[8] Tsankov P, Dan A, Drachsler-Cohen D, Gervais A, Buenzli F, Vechev M. Securify: Practical security analysis of smart contracts. In: Proc. of the 2018 ACM SIGSAC Conf. on Computer and Communications Security. Toronto: ACM, 2018. 67–82. [doi: 10.1145/3243734.3243780]

[9] Qu C, Tao M, Zhang J, Hong XY, Yuan RF. Blockchain based credibility verification method for IoT entities. Security and Communication Networks, 2018, 2018: 7817614. [doi: 10.1155/2018/7817614]

[10] Luu L, Chu DH, Olickel H, Saxena P, Hobor A. Making smart contracts smarter. In: Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security. Vienna: ACM, 2016. 254–269. [doi: 10.1145/2976749.2978309]

[11] Li PR, Li SS, Ding MJ, Yu JP, Zhang H, Zhou X, Li JY. A vulnerability detection framework for hyperledger fabric smart contracts based on dynamic and static analysis. In: Proc. of the 26th Int'l Conf. on Evaluation and Assessment in Software Engineering. Gothenburg: ACM, 2022. 366–374. [doi: 10.1145/3530019.3531342]

[12] Li JX, Wu JG, Chen L. Block-secure: Blockchain based scheme for secure P2P cloud storage. Information Sciences, 2018, 465: 219–231. [doi: 10.1016/j.ins.2018.06.071]

[13] Alharbi T. Deployment of blockchain technology in software defined networks: A survey. IEEE Access, 2020, 8: 9146–9156. [doi: 10.1109/ACCESS.2020.2964751]

[14] Fung CJ, Zhang J, Aib I, Boutaba R. Dirichlet-based trust management for effective collaborative intrusion detection networks. IEEE Trans. on Network and Service Management, 2011, 8(2): 79–91. [doi: 10.1109/TNSM.2011.050311.100028]

[15] Zhuang Y, Liu ZG, Qian P, Liu Q, Wang X, He QM. Smart contract vulnerability detection using graph neural networks. In: Proc. of the 29th Int'l Joint Conf. on Artificial Intelligence. Yokohama, 2020. 3283–3290.

[16] Afzaal H, Imran M, Janjua MU, Gochhayat SP. Formal modeling and verification of a blockchain-based crowdsourcing consensus protocol. IEEE Access, 2022, 10: 8163–8183. [doi: 10.1109/ACCESS.2022.3141982]

[17] Ribeiro M, Adão P, Mateus P. Formal verification of Ethereum smart contracts using Isabelle/HOL. In: Logic, Language, and Security: Essays Dedicated to Andre Scedrov on the Occasion of His 65th Birthday. Switzerland: Springer Int'l Publishing, 2020. 71–97. [doi: 10.1007/978-3-030-62077-6_7]

[18] Grishchenko I, Maffei M, Schneidewind C. A semantic framework for the security analysis of Ethereum smart contracts. In: Proc. of the 7th Int'l Conf. on Principles of Security and Trust. Thessaloniki: Springer, 2018. 243–269. [doi: 10.1007/978-3-319-89722-6_10]

[19] Krupp J, Rossow C. TEETHER: Gnawing at Ethereum to automatically exploit smart contracts. In: Proc. of the 27th USENIX Conf. on Security Symp. Baltimore: USENIX Association, 2018. 1317–1333.

[20] Feist J, Grieco G, Groce A. Slither: A static analysis framework for smart contracts. In: Proc. of the 2nd IEEE/ACM Int'l Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB). Montreal: IEEE, 2019. 8–15. [doi: 10.1109/WETSEB.2019.00008]

[21] Jiang B, Liu Y, Chan WK. ContractFuzzer: Fuzzing smart contracts for vulnerability detection. In: Proc. of the 33rd ACM/IEEE Int'l Conf. on Automated Software Engineering. Montpellier: ACM, 2018. 259–269. [doi: 10.1145/3238147.3238177]

© 中国科学院软件研究所 http://www.jos.org.cn

[22] Wang W, Song JJ, Xu GQ, Li YD, Wang H, Su CH. ContractWard: Automated vulnerability detection models for Ethereum smart contracts. IEEE Trans. on Network Science and Engineering, 2021, 8(2): 1133–1144. [doi: 10.1109/TNSE.2020.2968505]

[23] Norta A, Matulevičius R, Leiding B. Safeguarding a formalized blockchain-enabled identity-authentication protocol by applying security risk-oriented patterns. Computers & Security, 2019, 86: 253–269. [doi: 10.1016/j.cose.2019.05.017]

[24] Matsuo SI. How formal analysis and verification add security to blockchain-based systems. In: Proc. of the 2017 Formal Methods in Computer Aided Design (FMCAD). Vienna: IEEE, 2017. 1–4. [doi: 10.23919/FMCAD.2017.8102228]

[25] Chaudhary KC, Chand V, Fehnker A. Double-spending analysis of bitcoin. In: Proc. of the 24th Pacific Asia Conf. on Information Systems. Dubai, 2020. 210.

[26] Kalra S, Goel S, Dhawan M, Sharma S. Zeus: Analyzing safety of smart contracts. In: Proc. of the 2018 Network and Distributed Systems Security (NDSS) Symp. San Diego, 2018. 1–12. [doi: 10.14722/ndss.2018.23082]

[27] Geatti L, Gianola A, Gigante N. Linear temporal logic modulo theories over finite traces. In: Proc. of the 31st Int'l Joint Conf. on Artificial Intelligence. Vienna, 2022. 2641–2647. [doi: 10.24963/ijcai.2022/366]

[28] Ye X, Shi JQ, Huang YH, Li Q, Wei HS, Chen XY. Parallel computational tree logic model-checking on pushdown systems. Concurrency and Computation: Practice and Experience, 2022, 34(23): e7173. [doi: 10.1002/cpe.7173]

[29] Doostali S, Babamir SM, Javani M. Using a process algebra interface for verification and validation of UML statecharts. Computer Standards & Interfaces, 2023, 86: 103739. [doi: 10.1016/j.csi.2023.103739]

[30] Walkinshaw N, Hierons RM. Modelling second-order uncertainty in state machines. IEEE Trans. on Software Engineering, 2023, 49(5): 3261–3276. [doi: 10.1109/TSE.2023.3250835]

[31] Grossman S, Abraham I, Golan-Gueta G, Michalevsky Y, Rinetzky N, Sagiv M, Zohar Y. Online detection of effectively callback free objects with applications to smart contracts. Proc. of the ACM on Programming Languages, 2017, 2: 48. [doi: 10.1145/3158136]

[32] Schneidewind C, Grishchenko I, Scherer M, Maffei M. eThor: Practical and provably sound static analysis of Ethereum smart contracts. In: Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2020. 621–640. [doi: 10.1145/3372297.3417250]

[33] Shaikh E, Al-Ali AR, Muhammad S, Mohammad N, Aloul F. Security analysis of a digital twin framework using probabilistic model checking. IEEE Access, 2023, 11: 26358–26374. [doi: 10.1109/ACCESS.2023.3257171]

[34] Waldmann U, Tourret S, Robillard S, Blanchette J. A comprehensive framework for saturation theorem proving. Journal of Automated Reasoning, 2022, 66(4): 499–539. [doi: 10.1007/s10817-022-09621-7]

[35] Tempel S, Herdt V, Drechsler R. Specification-based symbolic execution for stateful network protocol implementations in IoT. IEEE Internet of Things Journal, 2023, 10(11): 9544–9555. [doi: 10.1109/JIOT.2023.3236694]

[36] Wang D, Dou WS, Gao Y, Wu CN, Wei J, Huang T. Model checking guided testing for distributed systems. In: Proc. of the 18th European Conf. on Computer Systems. Rome: ACM, 2023. 127–143. [doi: 10.1145/3552326.3587442]

[37] Khan KM, Arshad J, Khan MM. Empirical analysis of transaction malleability within blockchain-based e-voting. Computers & Security, 2021, 100: 102081. [doi: 10.1016/j.cose.2020.102081]

[38] Zhang SJ, Lee JH. Double-spending with a Sybil attack in the Bitcoin decentralized network. IEEE Trans. on Industrial Informatics, 2019, 15(10): 5715–5722. [doi: 10.1109/TII.2019.2921566]

[39] Shah Z, Ullah I, Li HL, Levula A, Khurshid K. Blockchain based solutions to mitigate distributed denial of service (DDoS) attacks in the Internet of Things (IoT): A survey. Sensors, 2022, 22(3): 1094. [doi: 10.3390/s22031094]

[40] Sahay R, Geethakumari G, Mitra B. A novel blockchain based framework to secure IoT-LLNs against routing attacks. Computing, 2020, 102(11): 2445–2470. [doi: 10.1007/s00607-020-00823-8]

[41] Tran M, Shenoi A, Kang MS. On the routing-aware peering against network-eclipse attacks in Bitcoin. In: Proc. of the 30th USENIX Security Symp. (USENIX Security 2021). 2021. 1253–1270.

[42] Du Y, Wang Z, Li J, Shi L, Jayakody DNK, Chen Q, Chen W, Han Z. Blockchain-aided edge computing market: Smart contract and consensus mechanisms. IEEE Trans. on Mobile Computing, 2023, 22(6): 3193–3208. [doi: 10.1109/TMC.2021.3140080]

[43] Wang JL, Liu Q, Song BY. Blockchain-based multi-malicious double-spending attack blacklist management model. The Journal of Supercomputing, 2022, 78(12): 14726–14755. [doi: 10.1007/s11227-022-04370-1]

[44] Hafid A, Hafid AS, Samih M. A tractable probabilistic approach to analyze Sybil attacks in sharding-based blockchain protocols. IEEE Trans. on Emerging Topics in Computing, 2023, 11(1): 126–136. [doi: 10.1109/TETC.2022.3179638]

[45] Piantadosi V, Rosa G, Placella D, Scalabrino S, Oliveto R. Detecting functional and security-related issues in smart contracts: A systematic literature review. Software: Practice and Experience, 2023, 53(2): 465–495. [doi: 10.1002/spe.3156]

[46] Qian P, Liu ZG, He QM, Huang BT, Tian DZ, Wang X. Smart contract vulnerability detection technique: A survey. Ruan Jian Xue

© 中国科学院软件研究所 http://www.jos.org.cn

Bao/Journal of Software, 2022, 33(8): 3059–3085 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6375.htm [doi: 10.13328/j.cnki.jos.006375]

[47] Dong WL, Liu Z, Liu K, Li L, Ge CP, Huang ZQ. Survey on vulnerability detection technology of smart contracts. Ruan Jian Xue Bao/Journal of Software, 2024, 35(1): 38–62 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6810.htm [doi: 10.13328/j.cnki.jos.006810]

[48] Han X, Yuan Y, Wang FY. Security problems on blockchain: The state of the art and future trends. Acta Automatica Sinica, 2019, 45(1): 206–225 (in Chinese with English abstract). [doi: 10.16383/jaas.c180710]

[49] Feng PH, Lian YF, Dai YX, Bao XH. A vulnerability model of distributed systems based on reliability theory. Ruan Jian Xue Bao/Journal of Software, 2006, 17(7): 1633–1640 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1633.htm [doi: 10.1360/jos171633]

[50] Gao HH, Zhang YD, Miao HK, Barroso RJD, Yang XX. SDTIOA: Modeling the timed privacy requirements of IoT service composition: A user interaction perspective for automatic transformation from BPEL to timed automata. Mobile Networks and Applications, 2021, 26(6): 2272–2297. [doi: 10.1007/s11036-021-01846-x]

[51] Cimatti A, Clarke E, Giunchiglia F, Roveri M. NuSMV: A new symbolic model checker. Int'l Journal on Software Tools for Technology Transfer, 2000, 2(4): 410–425. [doi: 10.1007/s100090050046]

[52] Hjálmarsson FÞ, Hreiðarsson GK, Hamdaqa M, Hjálmtysson G. Blockchain-based e-voting system. In: Proc. of the 11th IEEE Int'l Conf. on Cloud Computing (CLOUD). San Francisco: IEEE, 2018. 983–986. [doi: 10.1109/CLOUD.2018.00151]

[53] Marcilla B. Polygon-NFT-marketplace. 2022. https://github.com/ikcoin/Polygon-NFT-marketplace

[54] Lobo K. AuthentiFi. 2022. https://github.com/kylelobo/AuthentiFi

[55] Atraura Blockchain. scts. 2017. https://github.com/AtrauraBlockchain/scts

[56] yashverma9. AI blockchain electronic health records management system. 2020. https://github.com/yashverma9/AI-Blockchain-Electronic-Health-Records-Management-System

附中文参考文献:

[46] 钱鹏, 刘振广, 何钦铭, 黄步添, 田端正, 王勋. 智能合约安全漏洞检测技术研究综述. 软件学报, 2022, 33(8): 3059–3085. http://www.jos.org.cn/1000-9825/6375.htm [doi: 10.13328/j.cnki.jos.006375]

[47] 董伟良, 刘哲, 刘逵, 黎立, 葛春鹏, 黄志球. 智能合约漏洞检测技术综述. 软件学报, 2024, 35(1): 38–62. http://www.jos.org.cn/1000-9825/6810.htm [doi: 10.13328/j.cnki.jos.006810]

[48] 韩璇, 袁勇, 王飞跃. 区块链安全问题: 研究现状与展望. 自动化学报, 2019, 45(1): 206–225. [doi: 10.16383/jaas.c180710]

[49] 冯萍慧, 连一峰, 戴英侠, 鲍旭华. 基于可靠性理论的分布式系统脆弱性模型. 软件学报, 2006, 17(7): 1633–1640. http://www.jos.org.cn/1000-9825/17/1633.htm [doi: 10.1360/jos171633]

陈锦富(1978－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为软件测试, 软件安全, 可信软件.

施登洲(1997－), 男, 硕士, 主要研究领域为软件安全测试, 区块链漏洞检测.

冯乔伟(1998－), 男, 硕士生, CCF 学生会员, 主要研究领域为软件安全测试, 区块链漏洞检测.

Rexford Nii Ayitey SOSU(1986－), 男, 博士生, 主要研究领域区块链漏洞检测, 人工智能, 物联网, 云计算.

蔡赛华(1990－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为恶意流量检测, 异常数据检测, 软件安全测试.

© 中国科学院软件研究所 http://www.jos.org.cn