**中国科学技术大学**

**工程硕士研究生学位论文**

**开题报告**

**论文题目：Solidity智能合约源代码克隆检测**

**学 生 姓 名： 马明亮**

**学 校 导 师： 薛吟兴**

**企 业 导 师： 薛吟兴**

**工 程 领 域： 软件工程**

**领 域 代 码： 085212**

**研 究 方 向： 信息安全**

**所 在 院 系： 软件学院**

**实 习 单 位：中国科学技术大学苏州研究院**

**中国科学技术大学研究生院**

**填表日期： 2019 年 3 月 14 日**

**说 明**

1. **工程硕士学位论文的开题报告是保证论文质量的一个重要环节，为了加强对工程硕士研究生培养的过程管理，规范其学位论文的开题报告，特制此表。**
2. **工程硕士学位论文开题报告，应该在工程硕士学位授予点或培养单位组织的学术报告会上报告，听取意见，论证后再填写此表。**
3. **此表一式两份经导师和培养单位负责人签字后，交培养单位研究生教学管理办公室存档。**
4. **工程硕士研究生在申请学位论文答辩时，必须提交该学位论文开题报告。**

**一、 简况**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **研究生简况** | **学 号** | | **SA17225245** | | | | | | **姓 名** | **马明亮** | | | | **姓名拼音** | **Ma mingliang** | |
| **性 别** | | **男** | | **身份证号** | | | **411282199505300011** | | | | | | **出生年月** | **1995.5** | |
| **工程领域** | | **软件工程** | | | | | | | | | | **研究方向** | **软件安全** | | |
| **入学时间** | | **2017.9** | | | | **录取方式** | | | | **统考** | | | **培养方式** | | **全日制** |
| **本科毕业时间** | | | | | **2017．7** | | | | | | **本科毕业学校** | | **大连交通大学** | | |
| **论文类型与性质** | **名称** | **中文** | | **solidity智能合约源代码的克隆检测** | | | | | | | | | | | | |
| **英文** | | **Clone detection in SmartContract Solidity source code** | | | | | | | | | | | | |
| **类别** | **1．技术攻关研究 √ 2．工程项目策划**  **3．工程设计或技术改造 4．新工艺、新材料、新设备、新产品的研制与开发** | | | | | | | | | | | | | | |
| **形式** | **√1．工程设计**  **2．研究论文** | | | | | | | | | | | | | | |
| **性质** | **√1．****应用技术研究**  **2．技术开发** | | | | | | | | | | | | | | |
| **论文内容和意义** | **摘 要** | 智能合约是控制以太状态下账户行为的程序。允许相互不可信的实体在不依赖可信第三方的情况下进行交互。其中一个实现途径是启用智能合约，它通过编程编码规则来反映任何类型的多方交互。去年，区块链在上的投资超过了1亿4千万美元[16]，而且越来越多的人倾向于自主应用，智能合约正迅速成为实施金融工具(如货币、衍生品、钱包等)和应用的首选机制，尽管存在这样的潜力，但是不断出现的安全问题已经动摇了人们对智能合约的信任。一旦智能合约被部署到区块链上就很难再去修复自身的漏洞不管它管理多少资产。一个最近的漏洞导致了5000万美元加密货币的损失[15]同时经过我们的前期调研发现，在智能合约领域中代码的克隆现象十分普遍。也就是说如果一段智能合约程序存在一种漏洞，那么与之相似的一段目标代码很有可能也包含有这个漏洞。在这个特点下，这篇论文的主要内容就是如何将克隆检测算法与漏洞相结合。我们尝试运用克隆检测的方法去查找潜在漏洞。实现完全自动化系统的去检测智能合约是否是存在漏洞的。 | | | | | | | | | | | | | | |
| **主题词** | **主题词数量不多于三个，主题词之间空一格（英文用“/ ”分隔）** | | | | | | | | | | | | | | |
| **中文** | | **solidity克隆检测 漏洞** | | | | | | | | | | | | |
| **英文** | | **Solidity/ Clone Detection/ vulnerability** | | | | | | | | | | | | |

**二、选题依据**

|  |
| --- |
| 1. **阐述该选题的研究意义，或工程设计的价值和意义，国内外概况和发展趋势，选题的先进性和实用性，技术难度及工作量。**   **1.1选题的研究意义**  到2017年为止，整个加密货币市场资产已经达到了6000亿元[1]。工业界以及各国政府都被区块链的颠覆性“潜力”所吸引，区块链是加密货币的基础技术。加密货币使用区块链作为公共分类账，记录所有的货币转移，以避免重复花钱。虽然比特币是区块链技术最典型的应用，但除了加密货币，还有其他应用，如金融产品和服务、跟踪各种财产的所有权、数字身份验证、投票等。另一个热门话题是如何利用区块链技术来实现智能契约。抽象地说，智能合约是相互不信任的参与者之间的协议，它由区块链|的协商一致机制自动执行，而不依赖可信任的第三发权威机构。智能合约最突出的框架是以太坊，在以太坊中，智能合约被呈现为计算机程序，用图灵完整的语言编写。以太坊智能合约被正确执行是其有效性的必要条件:否则，对手可以篡改执行，例如从合法参与者那里转移一些钱给他自己。然而，仅仅执行的正确性并不足以保证智能契约的安全性。实际上，通过实际的开发经验和对以太坊区块链上所有合约的静态分析，已经发现了以太坊智能合约约中的几个安全漏洞。成百万的智能合约已经被部署在了各种各样的去中心化应用平台。在以太坊中已经有一些真正的攻击者利用这些漏洞从而导致了财产的损失。这些攻击中最成功的一次成功地从一份合同中窃取了约6000万美元[2]。Parity Wallet包含了两种漏洞，第一种漏洞损失了6000万美元，第二种漏洞冻结超过1500万美元[4][5]。有几个原因使得在以太坊中开发智能合约特别容易出错。其中有很重要的一个原因是智能合约使用solidity语言开发，solidity是一种以太坊支持的高级语言。solidity是一种图灵完备性型的编程语言允许开发者编写智能合约以及去中心化应用。以太坊生态环境正在快速成长到2018年的3月智能合约的数量和去中心化的应用已经增加到了两百多万[3]然而，根据智能合约管理如此多的财富，也使它们成为黑客攻击的诱人目标。  **1.2国内外概况和发展趋势**   * **智能合约漏洞检测国内发展现状**   在工业界当前国内还没有比较好的开源系统去对智能合约的漏洞进行检测。我们调研了腾讯的玄武实验室的智能合约安全检测平台并进行尝试发现检测的效果很不理想并且一次智能检测一个合约面对大型的合约文件是无法满足需求的，此外还有成都链安科技公司的智能合约检测需要付费才可以检测并且费用价格不菲。而且大多数智能合约漏洞检测是基于漏洞规则的检测或程序验证的技术等。这两个技术有一个共同的缺点是当面对一个新的漏洞出现时可扩展性不够。之后我们对玄武实验室平台进行了测试发现，检测的效果很不好。可见国内的对智能合约的漏洞检测还有很大的进步与提升空间。   * **智能合约漏洞检测国外发展现状**   越来越多地采用智能合约所以需要强有力的安全保障。不幸的是，创建没有安全漏洞的智能合约非常具有挑战性。因此，智能契约中的关键漏洞每隔几个月就会被发现和利用[6][7][8][9][10]。为了构建一个智能合约的安全检测器，最大的挑战就是solidity这门编成语言的图灵完备性。它使得自动任意属性的验证变得不可判定。为了解决这个问题，当前的自动化解决方案往往依赖于相当通用的测试和符号执行方法（例如 Oyente[11] and Mythril[12]）虽然这些方法在某些情况下很有用，但也有一些缺点:（1）它们可能错过重要的违规代码(由于近似的不足)，（2）而且容易产生误报(由于对领域知识的建模不足[13]),(3)他们可能无法在真实的合约场景中有效（Oyente在 Parity wallet上的有效覆盖率仅为20.2%[14]）也就是说检测的漏洞要么不太可能会出现要么不太可能被利用。总的来说，这些缺点给用户带来了很大的负担。他们必须检查所有报告是否有错误警报，还要担心未报告的漏洞。事实上，许多智能合约的安全属性天生就难以直接推理。  **1.3选题的先进性和实用性**  本系统是针对用solidity编写的智能合约进行漏洞检测，由于现如今的开源工具还没有用克隆技术对智能合约进行漏洞检测，另外智能合约的代码拷贝重用现象很普遍。所以我们提出用克隆检测方法去检测智能合约漏洞具有一定的先进性。  本系统是针对用solidity编写的智能合约进行漏洞检测，由于现如今的开源工具对智能合约的漏洞检测的效果都不好，而且随着加密货币的发展，智能合约的安全性问题日益重要。具有一定的实用性。  **1.4选题的技术难度和工作量**  技术难度：从源代码到IR的转换要根据不同的漏洞类型去抽象出不同的语义信息。  工作量：对solidity源代码进行分析构建出控制流图的过程具有一定的工作量.  **2.主要参考文献（列出作者、论文名称、期刊名称、出版年月）**   1. Bo Jiang. Bo Jiang, Ye Liu, and W.K. Chan. 2018. ContractFuzzer: Fuzzing Smart Contracts for Vulnerability Detection. In Proceedings of the 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE’18),2018/09 2. Analysis of the DAO exploit.http://hackingdistributed.com/analysis-of-thedao-exploit/.Last access,2018 3. Number of Smart Contracts on Ethereum. https://etherscan.io/Last access,2018. 4. The Parity Wallet Hack Explained.https://blog.zeppelin.solutions/.Last access,2018 5. The Wallet Smart Contract Frozen by the Parity Bug. https://github.com/paritytech/pari Last access, 2018 6. King of Ether. (2016). Available from: <https://github.com/kieranelby/KingOfTheEtherThrone/> 7. 2016. Etherdice. (2016). Available from: <https://etherdice.io/> 8. ]2016. theDAO. (2016). Available from: <https://etherscan.io/address/0xbb9bc244d798123fde783fcc1c72d3bb8c189413> 9. 2017.An In-Depth Look at the Parity Multisig Bug. (2017). Available from: <http://hackxingdistributed.com/2017/07/22/deep-dive-parity-bug> 10. Pawel Bylica. 2017. How to Find $10M Just by Reading the Blockchain. (Apr 2017). Available from: https://blog.golemproject.net/ how-to-fnd-10m-by-just-reading-blockchain-6ae9d39fcd95. 11. Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor.2016. Making Smart Contracts Smarter. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS). 254–269. 12. 2018. Mythril. (2018). Available from: <https://github.com/ConsenSys/mythril.> 13. Ilya Grishchenko, Matteo Maﬀei, and Clara Schneidewind. 2018. A Semantic Framework for the Security Analysis of Ethereum Smart Contracts. In Principles of Security and Trust - 7th International Conference (POST). 243–269. 14. 2018. Parity Wallet Library. (2018). Available from: <https://github.com/paritytech/parity/blob/4d08e7b0aec46443bf26547b17d10cb302672835/js/src/> 15. Zeus: Analyzing safety of smart contracts Kalra et al., NDSS’18 March 8, 2018 16. “Blockchain investment in 2016,” https://www.cryptocoinsnews.com/pwc−expert−1−4−billion −invested−blockchain−2016/ 17. DECKARD: Scalable and Accurate Tree-based Detection of Code Clones Lingxiao Jiang29th International Conference on Software Engineering (ICSE'07) |

**三、课题内容及具体方**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1. 课题内容**  本课题主要是针对solidity智能合约漏洞检测问题，而设计的一个系统，其主要包含以下4点内容：  （1）从源代码到抽象语法树构建。  （2）构建抽象语法树的特征向量。  （3）程序控制流图的构建。  （4）源代码到IR的转换，并对针对不同的代码漏洞类型对IR进行进一步的抽象。  （5）针对不同的漏洞类型选择不同的克隆检测技术。  （6）生成相似的报告。  **2、系统需求分析（应用软件工程专业描述工具描述）**  2.1 概述  一份明智的合同一旦部署就很难修补漏洞，无论它拥有多少资金。最近的一个漏洞造成了价值约5000万美元的加密货币损失。ZEUS评估大约94.6%的智能合约都存在安全隐患。虽然在一定程度上可以对合同的正确性进行手工审计，但仍然很费力，而且容易出错。另一方面，自动形式审计需要专门的工具和逻辑。与其他分布式系统代码不同，智能合约约是不可变的，并且在出现bug时很难修补，这一事实加剧了这个问题。例如，DAO的投资者损失了价值约5000万美元的加密货币，原因是代码中的一个漏洞允许攻击者反复重入取钱函数，导致财产的损失。经过调研分析发现智能合约程序的代码重用度非常高，将克隆检测方法用于智能合约漏洞的检测会是一个不错的选择。我们研发一款针对智能合约漏洞分析的克隆检测系统，来自动化的进行漏洞的检测。  2.2功能需求  solidity克隆检测系统用户主要是智能合约应用的开发者和对智能合约安全进行审计的专业人员。自动化的检测智能合约的潜在漏洞，减轻开发者和安全审计人员的负担。    图1 系统用例图  本需求着重描述克隆检测系统功能。此系统主要包括AST的信息提取、程序控制流图的构建、源代码到IR的转换与IR的语义信息的丰富与抽象、匹配算法的应用与调试。系统的工作时序图如图2所示。用户在命令行交互界面输入相应的检测命令，由命令解析器去判断用户输入的命令是否正确，如果命令正确就进行源代码的转化工作，然后转换器选择与漏洞对应的检测器执行检测，最后生成一份检测报告。    图2 系统时序图  2.3非功能性需求  1）运行环境：系统需要有solc0.4.25的编译器环境和python3环境。  2）交互需求：要为用户提供基本交互命令的说明。  **3、系统概要设计（应用软件工程专业描述工具描述）**  3.1 目标与概述  本设计将实现solidity克隆检测漏洞系统，系统将为智能合约的开发者和智能合约审计人员提供一份目标合约文件的漏洞检测报告。  本设计需要在设计完成后，能实现以下功能：  1. 用户可以单独指定某一种漏洞的detectors以提高检测效率。  2. 检测报告对每种漏洞尽量保证较低的FP和FN。  3. 本系统因为是应用克隆检测技术，所以要保证一定的可扩展性即当以后再找到其他类型的漏洞模板时可以很方便的添加一个detector就可以完善系统。保障系统一定的可扩展性。  另外，本设计在性能上要达到规模性、可扩展性和稳定性三项要求。规模性就是可以满足大量.sol文件的输入作为目标对象。可扩展性就是要求随着智能合约的发展新的漏洞类型会产生，我们可以直接添加相应的模板和detector就可以不断增强系统功能。稳定性就是当输入大批量待测文件时难免会有某些文件是不符合和语规范不能编译通过的文件，此时系统应该略过此文件继续下一个文件的检测保证系统的批处理特性。  3.2 系统总体设计设计  solidit源代码克隆检测系统主要是为用户提供一份目标文件的漏洞检测报告。所以系统的关键在于调研与分析已知漏洞的类型将这些漏洞提取出这些漏洞的签名，配合克隆检测技术来进行漏洞检测。克隆检测的分析是由三步组成：（1）大量阅读智能合约漏洞相关论文，提取与总结不同漏洞的基本源代码作为程序的匹配模板；然后构建solidity的抽象语法树与控制流图；（2）将solidity源代码转化成一种中间表示,以下统称IR.在编程语言设计中，编译器经常作用在一种中间表示（IR）.IR携带这个程序的额外的详细信息。通常情况下，编译器创建一个程序的解析树，该解析树表示已编写的程序。但是，编译器可以继续使用信息来丰富这个树，例如源代码所处的位置和其他可以影响控制流的项。除此之外，像solidity这样的语言具有继承性，这意味着函数和方法可以在合约的范围之外定义。IR可以使这些方法线性化，可以对源代码进行额外的转化和处理；（3）我们可以有多种方案来进行克隆检测，例如我们可以在抽象语法树基础上用树相似度匹配算法去进行模板程序和目标程序的匹配，我们也可以在控制流图或程序依赖图基础上运用图匹配算法进行匹配，同理也可以在IR的基础上运用串匹配算法进行克隆匹配检测。我们需要针对不同的漏洞特点，运用不同的克隆检测算法以求达到较好的漏洞检测效果。    图2 solidity克隆检测系统软件结构  solidity克隆检测系统的总体架构如图 2 所示。用户输入一个源代码文件系统提取源代码的抽象语法树信息，接着构建控制流图，然后将源代码转化成一种带有语义信息上的中间表达IR。最后我们可以根据各个不同的漏洞代码类型来决定选择在哪个基础上做克隆检测比较合适，在不同的基础上在语法树的基础上或控制流图的基础上或IR的基础上应用克隆算法来与我们程序中已经总结好的漏洞模板进行匹配。生成一份带有相似度的检测报告。  3.2.1 solidity克隆检测系统开发架构    图3 solidity克隆检测系统架构图  用户输入检测命令指定需要检测的漏洞类型，经过前期的调研根据同的漏洞类型选择效果最好的克隆算法，例如对于某些类型的DOS漏洞，与程序的控制流关系不大，我们就可以选择Token-based的串克隆匹配算法。但对于臭名昭著的DAO攻击事件。这个漏洞的主要原因是程序的控制流被外部合约控制，针对这个特性我们可以考虑用程序依赖图基础上来实现克隆检测，如图4是一个简化版的DAO漏洞代码。最后生成一份形似度的报告供用户进行参考。    图4 SimpleDAO  3.2.2 抽像语法树的构建架构  本系统的抽象语法树结构如图4所示，对程序进行解析生成抽象语法树。也可以将抽象语法树装换成JSON数据格式。然后对JSON进行解析，进行抽象语法树信息的提取与封装。    图5 抽象语法树结构图  3.2.3 控制流图的构建架构  本系统的控制流图结构，在AST的基础上我们对每一个函数中的每一条表达式进行封装成一个节点，根据程序的控制流来建立节点之间的前驱与后继。  3.3 克隆匹配算法模块设计  每种克隆检测技术都包含几个属性(我们也称之为维度)，通过这些属性可以明确特定的技术，例如，它是如何工作的，它做什么等等。将源代码的比较单元转换为另一种中间内部表示形式，以便进行比较或提取可比较的属性。这种转换可以非常简单，例如，只删除空格和注释。也可能比较复杂如PDG生成表示。或广泛的源代码转换。基于度量的方法通常从这种中间表示为每个比较单元计算一个属性向量。我们会根据solidity智能合约不同漏洞的特点来选择不同的转化方案以方便后续的克隆匹配阶段。克隆检测主要分为2个阶段，第一个阶段是对源代码进行转换，第二阶段是相似度比较阶段。在第一阶段，将源文本转换为内部格式，从而允许使用更有效的比较算法。在随后的比较阶段真正的匹配检测开始执行，由于其核心作用，根据检测技术的内部格式对其进行分类是合理的。因此，我们使用内部源代码表示对不同的检测技术进行分类本系统的检测模块大致可以分为三类模块。第一类模块Token-based技术模块；第二类模块Tree-based技术模块，第三类模块PDG-based技术模块。以下分别介绍三种模块的设计思路。  3.3.1Token-based技术模块设计  一段程序被处理成一段token序列。通过扫描token序列来找代码的克隆。在我们的方法中，对转换后的Token序列进行克隆关系识别，以便提取出仅仅修改变量名称的克隆。也就是说按照字面来看这两段程序是不相似的但在转换后的Token序列是相似的。然而在这里我有些问题需要去注意。应该注意的是，某些类型的克隆似乎很难重写为共享代码，即使它们是相同的代码部分。例如有这样一个代码片段，从函数定义的中间开始，到另一个函数定义的中间结束。为了进行有效的克隆分析，我们的克隆检测技术自动识别和分离每个函数定义除此之外还有标识符的正则化问题以及相似度的度量值设定问题。例如，识别结构和转换名称需要了解编程语言的语法规则。因此，克隆检测系统的前端，即词法分析和转换，应该包括所有与语言相关的部分;然而，后端，即匹配检测，应该是独立于语言的。完成这个模块需要4步。第一步：词法分析，每一行源文件都被划分为与编程语言的词法规则相对应的标记；第二步:对源代码进行转换。这一步分为两个子步骤。（1）根据变换规则进行变换，例如根据转化规则Toen序列的增加，删除，修改等（2）参数的替换，与类型、变量和常量相关的每个标识符都被替换为一个特殊的Token 这种替换使具有不同变量名称的代码部分成为克隆对，同时注意要将从转换后的token序列到原始序列的映射信息存储起来，在稍后的格式化步骤中会被用到；第三步：匹配检测，将目标源文件和我们已知的模板文件进行token序列的匹配得到一个相似度值；第四步：生成克隆检测报告。Token-based克隆匹配检测阶段我们可以运用后缀树算法（参考图7）或最长公共子序列算法（LCS）进行相似度匹配。    图6 Token-based克隆检测过程图    图7 CDUASTCDUASS’T$的后缀树，大节点是根  3.3.2 Tree-based技术模块设计  在Tree-based方法中，程序语言被解析成抽象语法树(AST)Tree-based的克隆检测精度更高。然后使用一些树匹配技术在树中搜索相似的子树，并以克隆对或克隆类的形式返回相似子树的对应源代码。编译器生成器用于生成带注释的解析树(AST)，并通过特征度量比较其子树。基于抽象语法树的方法忽略关于标识符的信息(为了使变量名称上不同的代码在抽象语法树上显示相同)。为了在AST中找到克隆，可以通过比较生成的解析树或AST中的子树来确定子树的精确匹配或紧密匹配。完成这个模块,我们可以用特征向量，去估计AST的内在信息，然后运用Locality Sensitive Hashing (LSH)进行相似向量的聚类，整体工作流程如图8所示。我们可以分成一下步骤。（1）：解析器是由正式语法自动生成的；（2）解析器将源代码转化成抽象语法树；（3）对解析树进行处理，生成一组固定维向量如图9所示；（4）这些向量根据它们的欧氏距离进行聚类；（5）额外的后续处理生成克隆报告。    图8 Tree-based克隆检测流程图    图9源代码片段    图10 Tree-base特征向量构建示意图[17]  如图9所示，我们通过一个小例子来说明算法的主要步骤。这两段代码是用来做数组的初始化的。这两个代码段的解析树是相同的，因为代码只在标识符名称和常量值上有所不同。这棵解析树如图10所示，在这里我们引入特征向量来对抽象语法树结构信息进行描述，这一步是关键的一步。一棵子树的特征向量是欧几里得空间中的一个点<c1,...,cn>其中，每个ci表示子树中特定树模式出现的次数。例如我们让树模式成为解析树中的节点类型。并不是解析树中的所有节点都是捕获树结构信息的必要节点。很多是重复的例如他们的父节点。我们因此要去区分相关节点和不相关节点。图10中实体轮廓的节点表示是相关的节点，而虚线轮廓的节点则是不相关的。不相关的节点在我们的向量中没有关联的模式或维度。例如，特征向量的有序维数是相关节点的出现次数。如图10所示id, lit, assign e, incr\_e, array\_e, cond\_e, expr\_s, decl, for\_s。因此，以decl为根的子树的特征向量为<1, 1, 0, 0, 0, 0, 0, 1, 0>，因为有一个id节点、一个lit节点和一个decl节点。我们通过对解析树的后序遍历，将子节点的向量与父节点的向量相加，据可以生成父节点的特征向量。例如根为assign\_e的子树的特征向量<2, 1, 1, 0, 1, 0, 0, 0, 0>是由特征向量array\_e<2, 0, 0, 0, 1, 0, 0, 0, 0>,primary\_e<0, 1, 0, 0, 0, 0, 0, 0, 0>,assign e<0, 0, 1, 0, 0, 0, 0, 0, 0>三个特征向量相加得到的。用户还可以指定最小特征向量数值总和最小值，来抑制小型子树的向量;这有助于避免报告通常无意义的小型克隆。在图10中这个阈值设定为3，因此没有为根在incr e的子树生成向量。通过改变这个阈值，我们可以系统地发现大型克隆。上述技术只考虑解析树中具有相应子树的代码片段。然而，开发人员经常在更大的上下文中插入代码片段。周围节点的差异可能会阻止将父节点检测为克隆。为了识别这些克隆片段，我们使用第二阶段的特征向量生成，称为向量合并，对特定节点序列的向量进行相加，在这个阶段，需要一个滑动窗口沿着序列化的解析进行移动。选择合适滑动窗口的大小，以便合并后的向量包含足够大的代码片段。在图10中我们和合并了decl和cond\_e来得到向量<3, 1, 0, 0, 0, 1, 0, 1, 0>为了对代码片段的合并。选择树中的哪些节点进行合并非常重要;这些节点必须在克隆代码之间建立良好的边界，同时不经常包含大型子树。表达式树的根(很可能是复制粘贴的原子单位)通常是合并向量的好选择。我们将这些选择的节点称为可合并节点。在图10中，可合并节点是for语句的四个子节点。如果我们选择任何语句都是可合并的，那么整个for循环将被视为一个没有子序列的单元。在图10中我们还要求每个合并后的每个向量中元素相加的的值大于5。如果我们需要值为6，那么将只有两个合并向量，而不是3个，是decl向量和cond\_e向量合并以及cond\_e,incr\_e和expr\_s合并。在选择特征向量后，我们的算法将相似的特征向量通过欧氏距离聚类来检测克隆代码。这两个相似的代码片段，都有相同的特征向量<6, 2, 1, 1, 1, 1, 1, 1, 1>。我们就可以报告这两段代码是克隆代码。由于生成的向量数目较大，需要一种有效的聚类算法。  3.3.2 PDG-based技术模块  由于智能合约代码大都比较简短，而且相比于其他的检测方法基于PGD的检测方法的优点是可以检测非连续的代码克隆所以某些漏洞可能比较适用于PGD-based克隆检测技术。PDG-based方法通过考虑源代码的语义信息，比其他方法更进一步地获得高抽象的源代码表示形式。PDG包含程序的控制流和数据流信息，因此携带语义信息。一旦从一个主程序中获得一组PDGs，应用同构子图匹配算法寻找返回为克隆的相似子图。但是对于代码连续性的漏洞就不会很适用程序依赖图检测，可以挑选前面两个模块来进行检测。基于程序依赖图，如图11，图12。克隆检测大致流程为以下步骤：（1）PDG中的所有节点都根据它们的内容进行散列。具有相同哈希值的节点被分类为等价类节点中使用的变量和常量在散列之前转换为它们的类型名称。因此，即使变量名不同，由相同的程序语法元素生成的哈希值也是相同的；（2）从每个等价类中选择一对节点(r1,r2)，并确定包含r1和r2的相似子图对；（3）如果一对克隆(s1, s2),在步骤2中是一对相似的子图中,被归入另一对克隆(s1’, s2’) (s1⊆s10 & s2⊆s20),则从组中删除检测克隆对因为报告包含克隆对没有用；(4)克隆集由共享相同子图的克隆对组成（5）形成克隆检测报告。    图11 solidity代码片段    图12 程序依赖图  3.4 IR设计模块  IR是源代码的一种中间表示，通过IR我们可以对源代码进行抽象和语义丰富。为后面的克隆匹配算法奠定基础。通过将源代码转换成IR,可以更好的对智能合约进行分析。例如，solidity的语法将.push作为对数组的一个函数调用。这种语义的直接表示形式与普通函数调用没有什么区别。相比之下，IR将数组的push调用作为一个特定的操作。允许进一步分析对数组的访问及其对程序安全性的影响。此外，IR中的操作符具有层次结构，因此例如，在几行代码中你可以追踪所有的写入一个变量的操作，这使得编写精确的污染分析变得非常简单。除此之外系统可以通过转化IR构建更丰富的合约表示，并允许对潜在的漏洞进行更深入的分析，例如，回答“用户能否控制变量”这个问题，对于从静态位置发现更复杂的漏洞至关重要。系统将将以迭代的方式将信息从函数参数传播到程序状态从而捕获可能跨多个事务的信息控制流。这样，系统可以丰富大量信息。IR变量设计大致如下表1：   |  |  | | --- | --- | | 全局变量 | StateVariable | | 局部变量 | LocalVariable | | 常量 | Constant | | solidity变量 | SolidityVariable | | 临时变量 | TemporaryVariable | | 引用变量 | ReferenceVariable | | 左值 | StateVariable, LocalVariable, TemporaryVariable, ReferenceVariable | | 右值 | tateVariable, LocalVariable, Constant, SolidityVariable,ReferenceVariable | | 赋值 | LVALUE := RVALUE | | LVALUE := Tuple | | LVALUE := Function | | 二元操作 | LVALUE = RVALUE \* RVALUE等 | | 一元操作 | LVALUE = ! RVALUE | | LVALUE = ~ RVALUE | | 索引 | EFERENCE -> LVALUE [ RVALUE ] | | push操作 | PUSH LVALUE RVALUE | | PUSH LVALUE Function | | 删除操作 | DELETE LVALUE | | 转换操作 | CONVERT LVALUE RVALUE TYPE | | 数组初始化操作 | LVALUE = INIT\_VALUES | | 调用操作 | LVALUE = HIGH\_LEVEL\_CALL DESTINATION FUNCTION [ARG, ..] | | LVALUE = LOW\_LEVEL\_CALL DESTINATION FUNCTION\_NAME [ARG, ..] | | 返回操作 | RETURN RVALUE，RETURN None | | 条件 | CONDITION RVALUE | | 删除操作 | DELETE LVALUE |   表1：IR转化大致规则  有了从源代码到IR的转化之后，相比较源代码我们得到了更加抽象的信息。在此基础上我们继续根据各种漏洞的特点进一步的去丰富IR的语义信息。最后尝试运用后缀树算法或者最长公共子序列的方法去比目标代码与漏洞模板代码IR之间的相似度。生成一份相似度报告。  **4、拟采用的设计方法、环境等**  利用solc0.4.25编译器生成的JSON格式的AST在此基础上抽取和封装大量程序结构信息  从AST构建出控制流图再转换成IR。  根据不同的漏洞类型选择不同的克隆检测算法。  开发环境为Pycharm,数据集为从Etherscan爬下来的合约18714个合约文件。  **5. 技术难度及特色分析**  技术难度  1.控制流图的构建过程。  2.从源代码到IR转换规则的指定  3.根据漏洞类型选择较为合适的克隆检测算法  本系统拥有很多特色   1. 克隆方法检测漏洞，具有良好的可扩展性。 2. 与动态分析相比较，克隆检测方法具有良好的时效性。 3. 将克隆检测算法应用到智能合约漏洞探测。   **6、本人主要工作描述**  分析已知的solidity智能合约的漏洞，并总结出漏洞模板作为克隆检测的匹配目标。从json格式的AST中提取封装程序信息，从而构建出控制流图。对控制流图中的节点内容进行IR的转换。分析漏洞特点选择较为合适的克隆匹配算法去计算目标文件和模板之间的相似度，最后生成相似度报告。 |
|  |

**四、工作进度的大致安排**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **应包括文献调研，工程设计，新工艺、新材料、新设备、新产品的研制和调试，实验操作，实验数据的分析处理，撰写论文等。**   |  |  |  | | --- | --- | --- | | 时间 | 工作 | 阶段成果 | | 2018.10 - 2018.11 | 完成对solidity智能合约漏洞调研，建立漏洞模板分析项目的可行性。 | 可行性分析报告，市场调研，开题报告 | | 2018.11 - 2018.12 | 完成前期课题调研，研究分析相关技术，并掌握项目总体框架的设计 | 需求分析报告 | | 2019.1 - 2019.3 | 掌握项目各模块具体的原理以及自己相关工作的重点和难点，熟悉系统架构的每个细节 | 概要设计报告 | | 2019.4 - 2019.6 | 根据设计文档完成自己相关模块的工作的详细设计，部署好开发环境，完成代码框架的编写 | 详细设计报告 | | 2019.7 - 2019.9 | 完成项目编码与测试工作 | 项目代码 | | 2019.10 - 2020.11 | 整理相关文档，撰写毕业论文，进行毕业论文答辩 | 毕业论文初稿 | | 2019.12 | 修改和完善毕业论文 | 毕业论文定稿 | |

|  |
| --- |
| **预期成果**   1. 完成solidity智能合约源代码克隆检测系统的设计与实现 2. 撰写工程硕士学位论文 3. 论文：争取发布CCF论文一篇 |
| **导师意见**  **导师签名：**  **年 月 日** |
| **培养单位负责人意见**  **培养单位负责人签名：**  **年 月 日** |