

硕 士 研 究 生 读 书 报 告



题目 智能合约漏洞检测技术学习与分析

作者姓名 李粤海

作者学号 22151206

指导教师 程学林

学科专业 软件工程

所在学院 软件学院

提交日期 二〇二二年一月

Study and Analysis of smart contract vulnerability detection technology

A Dissertation Submitted to Zhejiang University in partial fulfillment of the requirements for the degree of Master of Engineering

Major Subject: Software Engineering

Advisor: Cheng Xuelin

By

Li Yuehai

Zhejiang University, P.R. China

2022

**摘要**

随着区块链2.0时代的来临，智能合约作为一种图灵完备的编程语言被广泛地应用于区块链系统中。目前主流的智能合约语言是solidity语言，但由于其在许多层面仍不够成熟，因此存在许多漏洞容易被黑客攻击。本文首先简单介绍了智能合约及其特点，然后介绍其主要的漏洞类型，最后介绍了静态漏洞检测方法、动态漏洞检测方法和深度学习漏洞检测方法。

**关键词**： 智能合约；solidity；漏洞检测

**Abstract**

With the advent of the blockchain 2.0 era, smart contracts are widely used in blockchain systems as a Turing-complete programming language. The current mainstream smart contract language is the solidity language, but because it is still immature at many levels, there are many vulnerabilities that can be easily attacked by hackers. This article first briefly introduces smart contracts and their characteristics, then introduces their main types of vulnerabilities, and finally introduces static vulnerability detection methods, dynamic vulnerability detection methods, and deep learning vulnerability detection methods.

**Keywords:** smart contract; solidity; vulnerability detection

1 简介

智能合约使用的是Solidity语言，编写合约的过程实际上就是在构造一个类。首先，智能合约需要申明Solidity的版本，因为随着版本的升级会摈弃一些过时的表达。其次，在合约中需要申明变量和函数，这相当于提供接口给其他节点调用该合约。最后，每一个合约都会隐式地生成一个fallback()函数，这是为了避免外部节点或其他合约节点调用本合约时找不到合适的函数进行匹配。

每一个智能合约都需要由一个账户节点进行发布。当发布成功后，这个合约将会自己享用一个独立地址，而不受任何节点的掌控。因此，合约就如同法律一般，所有人都可以查看，且不受任何人的专制。当其他节点查看了该合约的源代码并认可该合约内容后，他们就可以通过调用合约中的函数进行合约内容的交互。

* 1. 智能合约的特点

绝大多数的智能合约都继承自它底层的区块链特点，主要包括安全性、透明性、实时性、自动化和精确性。安全性是指智能合约中使用了加密算法，因此它可以防止作恶方篡改记录。透明性是指当智能合约在公网上运行时，所有人都可以看到智能合约的内容以及分析它的用途。实时性是指多方对智能合约的操作几乎在同一时间起效。自动性是指智能合约一旦部署上链，就可以在任何时间发起对其的调用，它将会自动执行。精确性是指它不像一般合约具有灰色地带，用代码表达的逻辑都是很精确的。

1. 智能合约常见漏洞

以太坊问世不到十年，智能合约却屡次因自身逻辑漏洞而遭到黑客的攻击。这其中包括重入攻击、算法上下溢出攻击、未检查返回值、时间戳依赖和交易顺序依赖、任意存储写入攻击等，更详细的漏洞介绍说明和举例可以参照swc标准，这也是许多智能合约漏洞检测工具所参照的标准漏洞库。

* 1. 重入攻击

我们通过介绍最为轰动的The Dao众筹合约中的重入攻击，揭示合约代码的易受攻击性。这个众筹合约本来是对新型融资的一次伟大尝试，它在不到一个月的时间就融资超过了1亿美元。但是由于黑客通过反复调用fallback()函数实现无限制取款，就此宣告了该合约的破产，并迫使以太坊开发团队不得不以硬分叉的形式回滚这个合约。由此可见，在发布一个合约之前，必须要进行全方位的安全检查，否则一旦发布上链就难以更改。下面我们给出The Dao中的漏洞以及黑客攻击函数：

图形用户界面, 文本, 应用程序

描述已自动生成

图1.The Dao中的安全漏洞

文本

中度可信度描述已自动生成

图2.黑客合约中的fallback函数

如图2，这是The Dao中的安全漏洞。这里对应的功能是用户向The Dao索要其存在合约上的资金。The Dao则是先向用户转账，再将用户存在The Dao上的资金重置为0。如图2-5，这是黑客编写的合约中的fallback函数。在黑客合约中，仅有fallback是payable的，也就是说当黑客合约向The Dao发起取回存款的请求时，The Dao的退回存款的操作会触发fallback函数。我们注意到，fallback的内容就是在一定的栈深度和汽油费限制内，反复地调用The Dao的退回存款操作。而恰恰因为The Dao是先执行退款，后将资金置为0的，这样每次fallback都能从The Dao中盗取一笔等额的资金，直到The Dao上所有的资金被清空。因此，这里黑客成功对The Dao发起了循环调用，从而完成了重入攻击。于是，针对退回资金的函数编写，我们借此得到了一个教训：一定要先将资金余额置为0，再将钱转到用户账户上。

* 1. 整数上下溢出攻击

当将超出范围的数据存储到某数据类型中，会发生整数溢出的危险，而这种漏洞容易造成数字货币蒸发、高卖低收等事故。目前可采用OpenZeppelin提供的SafeMath库进行安全化运算。一个典型的案例是美链被整数溢出攻击：

文本

描述已自动生成

图3.整数溢出合约

我们可以看到amount的计算是存在溢出的风险的，例如当\_value足够大的时候，amout可能刚好溢出为0。这就会出现在转账一笔大额交易后，该用户的账户并没有扣除转账的前，从而该用户可以将合约中的钱全部盗走。

* 1. 未检查返回值

在evm机制中，底层调用例如send、delegate和call函数发生异常时，不会显式地抛出异常并进行回滚，它们仅仅只会返回一个false，因此调用者无法知道是否发生调用异常。因此需要特别的检查方法的返回值来捕获可能的异常。

2.4 时间戳依赖和交易顺序依赖

由于区块链中的一些属性可以由矿工控制，比如交易的顺序、区块生成的时间等，因此如果合约中使用了这些参数就有可能遭到矿工的攻击。例如，合约中通过时间戳来生成随机种子，由于矿工有权在一定区间内设置区块时间戳，这就有可能让他们找到有利的时间戳。又如，矿工可以根据gas的价格决定交易的打包顺序，通过监控交易池，攻击者就可以创造更高gas的价格使得自己的交易具有处理优先权。

2.5 任意存储写入攻击

如果一个合约中使用了动态数据结构，而未对该数据结构的索引值进行正确的限制的话，就有可能会发生任意存储写入漏洞：

文本

描述已自动生成

图4.任意存储写入漏洞

如图4所示，首先在PopBonusCode函数中会发生整数下溢，使得bonusCode动态数组的长度变为2^256-1。根据evm的存储策略，对象和具体的值存储的地址存在一个哈希映射关系。例如，bonusCode数组的长度存储在0号位置，它的值则存储在keccak256(0x0)的位置，同理owner对象存储在1号位置，它的值则存储在keccak256(0x1)的位置。由于evm的地址最多到只有2^256 – 1，而bonusCode数组始址为keccak256(0x0)，因此数组的第2^256 – keccak256(0x0) + 1即可指向owner对象的存储位置。因此，就可以通过UpdateBonusCodeAt函数对owner进行修改，从而实现权限的转换。

3 常用的漏洞检测技术

由于区块链的不可篡改性，智能合约一旦发布上链就难以更改，因此在上链前对智能合约进行全面的漏洞分析很有必要。一般而言，主流工具通过收集整理现存的智能合约漏洞，归纳出一个较全面细致的漏洞库，并利用静态分析方法和深度学习结合的漏洞检测方法对solidity智能合约给出漏洞分析报告。最新检测工具利用静态漏洞分析结合的深度学习模型对多种漏洞类型完成自动化检测，为了进一步实现漏洞的高覆盖率以及高准确率。同时，动态漏洞分析通过遍历所有的输入参数来捕获可能的异常，但由于巨大的输入空间和输入选择策略，导致耗时较长且存在一定的漏报情况。

3.1 静态漏洞检测

目前主流的静态漏洞检测工具通过研究智能合约编译后的字节码以及其对应的操作码，提取每种漏洞类型对应的操作码规则，利用符号执行/中间表示工具，得出已有漏洞类型的判断。同时，对于符号执行路径选择中，定义操作码中的高危指令，提高搜索效率。开源工具包括Oyente,Securify,Mythril,ContractGuard,Smartcheck等。

Mithril是以太坊官方的智能合约检测工具，主要步骤如下

1、获取合约字节码  
2、初始化合约账户的状态  
3、利用n个交易来探索合约的状态空间，n默认为2，但可以设置为任意值  
4、当发现不希望的状态时，证明或否定其在特定假设下的可到达性

模块1：反编译模块，将solidity代码或字节码转化为evm操作码

模块2：符号解释器模块，参考laser-ethereum工具。给定若干初始化合约账户，符号解释器将返回合约运行后的一系列状态（如图5），包括变量状态、执行环境状态等。Laser通过将合约拆分成一系列控制流图来分析合约的运行。

文本

低可信度描述已自动生成

图5.laser对应的状态

模块3：约束求解模块。利用laser，我们可以将合约拆分成一系列的状态和路径约束（控制流图）。Laser利用定理证明工具z3来检查路径约束的可满足性。

图示

描述已自动生成

图6.控制流图约束求解

模块4：漏洞库模块，通过建立统一标准规范的漏洞库，对满足某些特定匹配要求的漏洞进行报告反馈

模块5：漏洞模型模块，对不同种类的漏洞，提取出其状态或操作码中符合的规律，以便后续进行模型匹配

文本

描述已自动生成

图7.任意跳转漏洞

在漏洞匹配模块中，主要是针对目前的状态和约束面利用get\_transaction\_sequence函数尝试找到一个可达的交易序列，从而证明确实可以实现某个漏洞的条件。

3.2 动态漏洞检测

动态漏洞检测主要利用模糊测试，通过大面积覆盖函数参数的各种取值，从而达到检测各种异常输出的可能。如何减少无效测试用例的生成是提升效率的关键，例如可以在选择测试样例中使用动态污点技术和遗传算法。此外，动态检测还有一个优点是可以检测出新的漏洞。但由于其检测的复杂性，其开源程度较低，开源工具可参考ContractFuzzer。

3.3 机器学习/深度学习漏洞检测

随着人工智能的星期，基于深度学习和机器学习的漏洞检测技术逐渐被提出。基于深度/机器学习的合约漏洞检测方法依赖于统一和全面的智能合约漏洞数据集（或通过整合其他检测工具的结果），这种方法可以有效地提高漏洞检测的准确率，但存在可解释性差的问题。开源工具参考SaferSC,TMP等。

模块1：数据集整合模块，建立智能合约数据库，在网上获取大量智能合约，将其存入数据库

模块2：标签模块，通过已有的漏洞检测工具结合人工判断给数据集中的智能合约标记是否满足某些漏洞

模型3：特征提取模块，通过定义操作码的前后依关系（一种机器学习特征提取）或将代码转化为控制流图进而转化为点集和边集（深度学习工具TMP）的方法，获取每个合约的特征向量

模块4：模型构建与选择模块，通过选取多个机器学习/深度学习模型，划分训练集和测试集，遍历超参数并对各个性能指标进行记录，选取出表现最优的模型和参数

优点：准确率往往更高，耗时很短

缺点：检测范围一般较窄（各漏洞的训练样本数量差别大）

3.3.1 特征提取

在机器学习/深度学习方法中，最重要的两步是分别是从智能合约中提取特征，以及训练出最优的超参数和模型。其中，最关键的还是提取特征步骤，因为参数和模型的选择tensorflow或pytorch已经有较好的封装，但特征的提取是需要完全人工操作的。我们主要介绍一种基于操作码的特征提取和一种基于图的特征提取。

基于操作码的特征提取：

表格

描述已自动生成

图8.基于操作码的特征提取

首先，我们需要对solidity源代码或evm字节码转换成操作码，由于操作码的有限性我们更容易对其进行分析。我们得到操作码序列后（如图8），使用bigram策略考虑操作码任意相邻两个的依赖关系，并根据bigram出现的频率作为特征构成特征矩阵。最后将特征喂入训练模型即可。

基于图的特征提取：

图示

描述已自动生成

图9.基于图的特征提取

这种特征提取方法主要是将合约中重要的函数和变量、控制流数据流的信息转化为点和边的信息。后续，进一步通过节点的精简和边的消融得到节点和边的one-hot编码，并将其喂入训练模型中。

主要节点的构造：

主节点主要表示对自定义或内置函数的调用，这些往往是对检测具体漏洞起到重要作用的调用。例如，在重入漏洞中，一个主节点可以是对transfer函数的调用或者是内置call.value函数的调用，这两个都是侦察重入的关键。对时间戳依赖而言，内置调用block.timestamp将提取为一个主节点。对无限循环漏洞而言，所有合约内自定义的函数都被看成主节点。更正式来说，我们将关键的函数视为主节点，它们记为M1,M2…

次要节点的构造：

主节点表示的是重要的调用，那么次要节点就用来表示模型中重要的变量，例如用户的余额和是否奖励的标志。这些关键的变量将会被定义为次要节点，记为S1,S2…

fallback节点的构造：

更进一层地，我们通过构造fallback节点F来模拟fallback函数对合约进行攻击，它是可以与合约进行交互的。fallback函数是智能合约中的一个特殊的设计，也是许多安全漏洞中的原因。

边的构造：

我们通过构造边来描述模型中各节点的关系。每一条边都描述了一条路径，该路径可能会被测试的合约函数遍历，而这条边的暂时的数字表示了它在该函数里面的顺序。具体来说，一条边的特征被提取为一个元组： (Vs , Ve , o, t)，其中，Vs和Ve表示它的起始和结束节点，o表示它暂时的顺序，t表示边的类型。为了获取节点之间更丰富的语法依赖，我们构建了四种不同类型的边，分别是控制流、数据流、正常前向和fallback流边。

3.2 集成漏洞检测

鉴于动态漏洞检测普遍具有较高的漏报率和误报率，SCstudio通过整合四种静态分析工具在一定时间的分析结果得到一个更全面的漏洞分析报告。

图示, 文本

描述已自动生成

图10.SCstudio实现框架

同时，SCstudio作为插件在VScode中出现，提供了智能的选词建议列表以及实时的漏洞标亮分析报告。

图形用户界面, 应用程序, Word

描述已自动生成

图11.SCstudio漏洞分析报告

作为一种集成的漏洞检测方法，SCstudio的实现较为简单且漏洞覆盖面更广。

4 国内外智能合约检测技术总结

就目前而言，几种主要类型的漏洞检测方法包括静态检测中的符号执行和动态检测中的模糊检测都是以国外研究的漏洞检测系统占优。在语义分析方法中，我国的趣链、成都链安公司都开发出一套精度颇高的检测系统；同时在深度学习检测方法中，浙江大学的团队也做出具有突破性的贡献。

因此，为了我国更好地发展合约漏洞检测技术，应该继续加强静态、动态和深度学方法在各漏洞的结合研究，从而达到更高的漏洞检测精度和更广的覆盖率！

参考文献

[1]孙家泽, 余盼盼, 王小银,等. 智能合约漏洞检测技术研究综述[J]. 西安邮电大学学报, 2020, 25(5):10.

[2]杨坤. 基于符号执行的智能合约自动化安全审计[D].电子科技大学,2020.

[3]宋晶晶. 以太坊智能合约的自动化漏洞检测方法研究[D].北京交通大学,2020.

[4] Zhuang Y , Liu Z , Qian P , et al. Smart Contract Vulnerability Detection using Graph Neural Network[C]// Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20. 2020.

[5]Meng Ren, Fuchen Ma, Zijing Yin, Huizhong Li, Ying Fu, Ting Chen, and Yu Jiang. 2021. SCStudio: a secure and efficient integrated development environment for smart contracts. In <i>Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis</i> (<i>ISSTA 2021</i>). Association for Computing Machinery, New York, NY, USA, 666–669. DOI:https://doi.org/10.1145/3460319.3469078