(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112256271 A
(43)申请公布日 2021.01.22

(21)申请号 202011116748.X

(22)申请日 2020.10.19

(71)申请人 中国科学院信息工程研究所
地址 100093 北京市海淀区闵庄路甲89号

(72)发明人 王瑜 周启慧 王雅哲 汪晗 范洪端

(74)专利代理机构 北京科迪生专利代理有限责任公司 11251
代理人 张乾桢

(51)Int.Cl.
G06F 8/41(2018.01)
G06F 21/56(2013.01)
G06Q 40/04(2012.01)

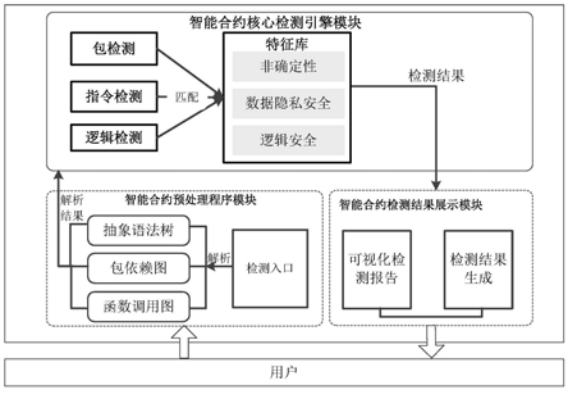
权利要求书3页 说明书7页 附图4页

(54)发明名称

一种基于静态分析的区块链智能合约安全检测系统

(57)摘要

本发明公开了一种基于静态分析的区块链智能合约安全检测系统，包括智能合约预处理程序模块、核心检测引擎模块和检测结果展示模块，通过智能合约预处理程序模块，对合约进行抽象语法树分析和内部调用关系分析，实现对合约函数关联关系的分析；核心检测引擎模块根据非确定性、数据隐私安全和逻辑安全每种风险项的特征，采用包括包检测、指令检测和逻辑检测等，分析得出合约的风险项；最后检测结果通过可视化展示模块呈现给用户。



CN 112256271 A

1.一种基于静态分析的区块链智能合约安全检测系统，包括智能合约预处理程序模块、核心检测引擎模块和检测结果展示模块，其特征在于：

智能合约预处理程序模块：用于验证智能合约源代码的静态语法合法性，包括：验证不对称代码块括号错误、结构体标签格式错误、无效代码错误，验证通过后将合约源代码解析为一系列静态结构，包括抽象语法树、包依赖关系、函数调用关系，为后续智能合约核心检测引擎模块提供基础分析数据；

智能合约核心检测引擎模块：用于首先把每个合约安全风险项抽象为静态结构特征形成特征库，包括非确定性特征库、数据隐私安全特征库和逻辑安全特征库；其次采用包检测、指令检测和逻辑检测将预处理程序模块所得到的源代码静态结构与特征库检测匹配；最后根据匹配情况得到合约合法性检测结果；

智能合约检测结果展示模块：用于将合约核心检测引擎模块获得的检测结果的合法性，生成可下载至本地的合约检测报告并将检测结果可视化，以及在线查看有风险项的源代码文件以及高亮显示的风险代码行，提供直观的检测结果展示。

2.根据权利要求1所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于：

智能合约核心检测引擎模块语法检验合法性后，采用基于抽象语法树分析法和内部调用关系分析法对智能合约源代码进行静态结构分析，获得智能合约抽象语法树、包依赖关系图和函数调用关系图等静态结构；所述内部调用关系分析法包括包依赖关系分析法和函数依赖关系分析法。

3.根据权利要求2所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于：

抽象语法树分析法采用基于合约编译器的词法分析与语法分析，完成对合约源代码语法树的解析，其中词法分析是对源代码的字符串序列解析，并转换成词语序列；语法分析基于语法库并根据合约语法规则定义，将词语序列转化为抽象语法树结构AST；该智抽象语法树结构以整个文件为根节点，自上而下描述文件中不同层级的语法结构。

4.根据权利要求2所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于：

所述包依赖分析法采用基于中间码解析获得合约源代码所引入的包间依赖关系，中间码生成过程即由抽象语法树翻译到中间码的过程，基于通用语言中间码的静态单赋值特性，使得生成的包依赖逻辑关系清晰可查；

整个分析过程从智能合约源程序开始，源程序视为最顶层的包，自上而下依次递归地读取依赖包信息，构建链码的包依赖关系，其中，level为当前依赖层数，maxLevel为设置的依赖最大层数；在构建智能合约包依赖关系时，分析包括智能合约源代码开始的3层包依赖关系，即maxLevel设置的依赖最大层数为3。

5.根据权利要求4所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于，所述分析包依赖关系进一步包括：

a.加载智能合约源代码，设置level当前依赖层数为0，maxLevel依赖最大层数为3，记录为智能合约依赖包的根节点；

b.提取通用语言中间码中的引用import，设置level当前依赖层数为1，并记录当前根

节点的子节点N；

c.依次递归地对各个包引用import解析依赖，判断level是否小于maxLevel，若小于maxLevel值则将level当前依赖层数自加1，并记录当前子节点的子节点N'，并继续执行c；若大于maxLevel值则执行结束，获得最终的包依赖关系图。

6.根据权利要求4所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于：

所述函数依赖分析借助源代码的中间码的静态单赋值特性，根据基于包含的指针分析构建函数调用关系，并对生成的函数调用关系进行筛选，得到智能合约中函数调用关系图；具体的，

首先基于通用语言的指针库构建最原始基于中间码的函数调用图Callgraph；其次采用深度优先的方式遍历原始调用图Callgraph中每条调用边，按照对后续安全检测有用规则，判断每个调用边上的调用函数Caller与被调用函数Callee是否满足规则，若调用边符合规则，则将该调用边上的调用函数Caller与被调用函数Callee添加到函数调用关系图集合中，依次重复执行构造出智能合约内部全部的函数调用关系图。

7.根据权利要求1所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于：

所述智能合约核心检测引擎模块，采用包检测首先通过深度遍历算法对包依赖关系图进行检测搜索，判定引用import关键字所定义的标准库或外部包是否在特征库的黑名单、建议库和忽略库内；其中黑名单包括标准库黑名单和接入外部库黑名单，建议库包括如crypo/md5，crypo/des，忽略库包括以github.com/hyperledger为前缀的库；

其次，对于检测到黑名单中的包引用，判断包引用所处依赖的层数；若该包引用处在包依赖关系图的第三依赖层，则认为该包引用与智能合约的逻辑关联性已经不大可直接忽略；若该包引用处在包依赖关系图的第一、二依赖层，则判断存在安全风险并标注；若检测到包引用处在包依赖的第一层且未使用建议库中的包，则给出意见询问是否需要加密让数据更加安全；

最后，收集整理检测结果反馈给可视化检测结果展示模块，该包检测主要完成对随机数生成、获取系统时间、访问文件系统或执行命令、引入第三方库和未对敏感数据加密安全风险项的检测。

8.根据权利要求1所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于：

所述智能合约核心检测引擎模块采用指令检测，首先通过树深度遍历算法对抽象语法树进行检测搜索，检测抽象语法树节点特征是否与特征库匹配来判定风险项及定位风险项；其次，对与检测到抽象语法树节点特征与特征库匹配，则该节点对应的源代码存在安全风险并标注，包括：通过读取全局变量、智能合约结构体中声明变量、遍历Map结构、并发程序等在语法树中变量、结构体、range语句以及节点信息特征，判断所属风险及定位风险位置；通过读取对链调用、隐私数据存取等函数节点的信息，判断所属风险及定位风险位置；最后，收集整理检测结果反馈给可视化检测结果展示模块；该指令检测主要完成对定义全局变量、合约结构体中声明变量、遍历Map结构、程序并发、跨链合约调用和未利用隐私数据机制的安全风险项的检测。

9.根据权利要求1所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于：

所述智能合约核心检测引擎模块采用逻辑检测，首先通过深度遍历算法对函数调用关系图进行检测搜索函数调用路径特征是否与特征库匹配来判定风险项及定位风险项；其次，对与检测到函数调用路径特征与特征库匹配，则该节点对应的源代码存在安全风险并标注，如不可重复执行的数据查询检测需要在函数调用关系图中，确认是否存在由反射函数Invoke到数据范围查询函数的调用路径；若路径存在，则判定存在该风险项及定位风险项；读写不一致风险也同样检测函数调用关系图中，对数据状态的读取以及写入操作生成“反向调用关系树”，树结构中存在相同父节点，判定存在该风险项及定位风险项；最后，收集整理检测结果反馈给可视化检测结果展示模块；该逻辑检测完成对不可重复执行的数据查询和读写不一致的安全风险项的检测。

10.根据权利要求1所述的一种基于静态分析的区块链智能合约安全检测系统，其特征在于：

所述可视化检测结果展示模块，首先通过分布式WEB节点组成基于区块链的分布式展示网络，将从合约核心检测引擎模块获得的检测结果通过共识机制存储在分布式展示网络账本共享；其次，通过编写智能合约实现分布式的下载至本地的合约检测报告并将检测结果可视化；最后，将展示结果的所有操作记录存储在分布式账本上，保证可审计追溯。

一种基于静态分析的区块链智能合约安全检测系统

技术领域

[0001] 本发明属于信息安全中的安全检测技术领域，具体涉及一种为基于静态分析的区块链智能合约安全检测系统。

背景技术

[0002] 随着比特币、以太坊的广为人知，区块链被视为一种强大的技术，广泛地影响着多个领域，越来越多的研究机构关注区块链技术的演进以及在不同应用场景下的实际落地，如金融货币、供应链、身份认证等。智能合约则是运行在区块链之上的核心构成要素之一。智能合约本质是一段计算机程序，通过区块链网络上的事件触发，实现非信任方之间的共识在满足一定条件下自动执行。智能合约将改造传统的业务流程，实现可靠安全地控制和管理链上传递的数据、资产等。目前基于联盟链框架智能合约正处于上升趋势，合约开发趋向于与上层应用的结合，在全网催生了十分丰富的“去中心化”应用生态，基于Fabric的智能合约正在为多行业、大规模的应用尤其是商业化应用赋能。

[0003] 目前联盟架构的最具代表性Fabric支持多种当前流行的通用高级语言来开发智能合约，如Golang。Fabric支持通用语言的特性为开发人员提供了很大的逻辑处理灵活性，大大降低了开发的成本。通用语言在方便开发人员开发合约的同时，也淡化了合约与普通程序间的界限。由于通用语言并不是为智能合约而专门设计的，通用语言开发者可能忽略与智能合约机制相关的特性，又或者由于开发者水平不一而导致合约质量良莠不齐，容易使开发完成智能合约引入安全风险。而当前通用语言自身的开发和审计工具并不能识别所有与合约相关的安全风险。

发明内容

[0004] 本发明技术解决问题：本发明旨在为联盟架构的最具代表性Fabric智能合约提供安全检测，使得运行在区块链上的智能合约更加安全可靠。本发明基于静态分析的区块链智能合约安全检测系统，首先利用合约预处理程序，对合约文件解析其语法树、包依赖、函数依赖等信息，为后续操作提供基础内容；其次利用核心检测引擎，对Fabric智能合约三大安全风险的代码特征形成特征库，并设计包检测、指令检测、逻辑检测等方法，通过匹配特征库以确定风险项及其风险位置；最后通过前端可视化模块，向用户提供可视化的检测报告，包括风险项描述与风险位置，以及消除风险项的相关指导建议，从而及时发现合约安全问题，保证合约的安全性、可靠性。

[0005] 本发明技术解决方案：本发明通过智能合约预处理程序，对合约进行抽象语法树分析和内部调用关系分析，实现对合约函数关联关系的分析；核心检测引擎根据非确定性、数据隐私安全和逻辑安全每种风险项的特征，采用包括包检测、指令检测和逻辑检测等，分析得出合约的风险项；最后检测结果通过可视化展示模块呈现给用户。

[0006] 本发明提出一种基于静态分析的区块链智能合约安全检测系统，包括智能合约预处理程序模块、核心检测引擎模块和检测结果展示模块，包括：

[0007] 智能合约预处理程序模块:用于验证智能合约源代码的静态语法合法性,包括:验证不对称代码块括号错误、结构体标签格式错误、无效代码错误,验证通过后将合约源代码解析为一系列静态结构,包括抽象语法树、包依赖关系、函数调用关系,为后续智能合约核心检测引擎模块提供基础分析数据;

[0008] 智能合约核心检测引擎模块:用于首先把每个合约安全风险项抽象为静态结构特征形成特征库,包括非确定性特征库、数据隐私安全特征库和逻辑安全特征库;其次采用包检测、指令检测和逻辑检测将预处理程序模块所得到的源代码静态结构与特征库检测匹配;最后根据匹配情况得到合约合法性检测结果;

[0009] 智能合约检测结果展示模块:用于将合约核心检测引擎模块获得的检测结果的合法性,生成可下载至本地的合约检测报告并将检测结果可视化,以及在线查看有风险项的源代码文件以及高亮显示的风险代码行,提供直观的检测结果展示。

[0010] 进一步的,智能合约核心检测引擎模块语法检验合法性后,采用基于抽象语法树分析法和内部调用关系分析法对智能合约源代码进行静态结构分析,获得智能合约抽象语法树、包依赖关系图和函数调用关系图等静态结构;所述内部调用关系分析法包括包依赖关系分析法和函数依赖关系分析法。

[0011] 进一步的,抽象语法树分析法采用基于合约编译器的词法分析与语法分析,完成对合约源代码语法树的解析,其中词法分析是对源代码的字符串序列解析,并转换成词语序列;语法分析基于语法库并根据合约语法规则定义,将词语序列转化为抽象语法树结构AST;该智抽象语法树结构以整个文件为根节点,自上而下描述文件中不同层级的语法结构。

[0012] 进一步的,所述包依赖分析法采用基于中间码解析获得合约源代码所引入的包间依赖关系,中间码生成过程即由抽象语法树翻译到中间码的过程,基于通用语言中间码的静态单赋值特性,使得生成的包依赖逻辑关系清晰可查;

[0013] 整个分析过程从智能合约源程序开始,源程序视为最顶层的包,自上而下依次递归地读取依赖包信息,构建链码的包依赖关系,其中,level为当前依赖层数,maxLevel为设置的依赖最大层数;在构建智能合约包依赖关系时,分析包括智能合约源代码开始的3层包依赖关系,即maxLevel设置的依赖最大层数为3。

[0014] 进一步的,所述分析包依赖关系进一步包括:

[0015] a.加载智能合约源代码,设置level当前依赖层数为0,maxLevel依赖最大层数为3,记录为智能合约依赖包的根节点;

[0016] b.提取通用语言中间码中的引用import,设置level当前依赖层数为1,并记录当前根节点的子节点N;

[0017] c.依次递归地对各个包引用import解析依赖,判断level是否小于maxLevel,若小于maxLevel值则将level当前依赖层数自加1,并记录当前子节点的子节点N',并继续执行c;若大于maxLevel值则执行结束,获得最终的包依赖关系图。

[0018] 进一步的,所述函数依赖分析借助源代码的中间码的静态单赋值特性,根据基于包含的指针分析构建函数调用关系,并对生成的函数调用关系进行筛选,得到智能合约中函数调用关系图;具体的,

[0019] 首先基于通用语言的指针库构建最原始基于中间码的函数调用图Callgraph;其

次采用深度优先的方式遍历原始调用图Callgraph中每条调用边，按照对后续安全检测有用规则，判断每个调用边上的调用函数Caller与被调用函数Callee是否满足规则，若调用边符合规则，则将该调用边上的调用函数Caller与被调用函数Callee添加到函数调用关系图集合中，依次重复执行构造出智能合约内部全部的函数调用关系图。

[0020] 进一步的，所述智能合约核心检测引擎模块采用包检测，首先通过深度遍历算法对包依赖关系图进行检测搜索，判定引用import关键字所定义的标准库或外部包是否在特征库的黑名单、建议库和忽略库内；其中黑名单包括标准库黑名单和接入外部库黑名单，建议库包括如crypo/md5，crypo/des，忽略库包括以github.com/hyperledger为前缀的库；

[0021] 其次，对于检测到黑名单中的包引用，判断包引用所处依赖的层数；若该包引用处在包依赖关系图的第三依赖层，则认为该包引用与智能合约的逻辑关联性已经不大可直接忽略；若该包引用处在包依赖关系图的第一、二依赖层，则判断存在安全风险并标注；若检测到包引用处在包依赖的第一层且未使用建议库中的包，则给出意见询问是否需要加密让数据更加安全；

[0022] 最后，收集整理检测结果反馈给可视化检测结果展示模块，该包检测主要完成对随机数生成、获取系统时间、访问文件系统或执行命令、引入第三方库和未对敏感数据加密安全风险项的检测。

[0023] 进一步的，所述智能合约核心检测引擎模块采用指令检测，首先通过树深度遍历算法对抽象语法树进行检测搜索，检测抽象语法树节点特征是否与特征库匹配来判定风险项及定位风险项；其次，对与检测到抽象语法树节点特征与特征库匹配，则该节点对应的源代码存在安全风险并标注，包括：通过读取全局变量、智能合约结构体中声明变量、遍历Map结构、并发程序等在语法树中变量、结构体、range语句以及节点信息特征，判断所属风险及定位风险位置；通过读取对链调用、隐私数据存取等函数节点的信息，判断所属风险及定位风险位置；最后，收集整理检测结果反馈给可视化检测结果展示模块；该指令检测主要完成对定义全局变量、合约结构体中声明变量、遍历Map结构、程序并发、跨链合约调用和未利用隐私数据机制的安全风险项的检测。

[0024] 进一步的，所述智能合约核心检测引擎模块采用逻辑检测，首先通过深度遍历算法对函数调用关系图进行检测搜索函数调用路径特征是否与特征库匹配来判定风险项及定位风险项；其次，对与检测到函数调用路径特征与特征库匹配，则该节点对应的源代码存在安全风险并标注，如不可重复执行的数据查询检测需要在函数调用关系图中，确认是否存在由反射函数Invoke到数据范围查询函数的调用路径；若路径存在，则判定存在该风险项及定位风险项；读写不一致风险也同样检测函数调用关系图中，对数据状态的读取以及写入操作生成“反向调用关系树”，树结构中存在相同父节点，判定存在该风险项及定位风险项；最后，收集整理检测结果反馈给可视化检测结果展示模块；该逻辑检测完成对不可重复执行的数据查询和读写不一致的安全风险项的检测。

[0025] 进一步的，所述可视化检测结果展示模块，首先通过分布式WEB节点组成基于区块链的分布式展示网络，将从合约核心检测引擎模块获得的检测结果通过共识机制存储在分布式展示网络账本共享；其次，通过编写智能合约实现分布式的下载至本地的合约检测报告并将检测结果可视化；最后，将展示结果的所有操作记录存储在分布式账本上，保证可审计追溯。

[0026] 本发明与现有技术相比,具有以下显著优点:

[0027] (1)本发明通过基于静态分析的区块链智能合约安全检测使系统具有识别所有与智能合约自有特点相关的安全风险,保证合约后续运行的安全性、可靠性。

[0028] (2)本发明可为智能合约开发者提出可视化开发指导建议,为智能合约提供运行前的安全保障。

附图说明

[0029] 图1本发明的系统总体框架;

[0030] 图2基于抽象语法树和内部调用关系分析的智能合约预处理程序流程框图;

[0031] 图3基于Golang语言所开发Fabric智能合约的抽象语法树结构图;

[0032] 图4基于中间码解析的包依赖关系图生成流程图;

[0033] 图5基于中间码解析的包依赖关系示例图;

[0034] 图6基于中间码和深度边遍历解析的函数关系图生成流程图;

[0035] 图7基于包检测、指令检测和逻辑检测与特征库自动化匹配的智能合约核心检模型图;

[0036] 图8基于分布式WEB工作流结构的可视化检测结果展示模型图。

具体实施方式

[0037] 为使本发明的目的、优点以及技术方案更加清楚明白,以下通过具体实施,并结合附图,对本发明进一步详细说明。

[0038] 对于图1是本发明基于静态分析的区块链智能合约安全检测系统的系统框图,包括智能合约预处理程序模块、核心检测引擎模块和检测结果展示模块。概括来说,一是智能合约的开发者将合约源代码上传至程序预处理模块,生成代码静态结构;二是代码静态结构自动化输入核心检测引擎模块,采用特征匹配检测合约安全问题;三是基于分布式WEB工作流结构的将检测结果可视化呈现给开发者,安全规范的指导合约开发者进行合约开发。

[0039] 根据本发明的一个实施例,所述智能合约预处理程序模块用于基于抽象语法树和内部调用关系分析,如图2所示,该模块主要是完成对智能合约源代码的合法性进行检测,首先对开发者上传的打包后智能合约源代码zip压缩包进行包文件完整性检验;其次包完整性检验通过后,采用基于分析器列表的自定义语法检查,对包含的合约文件检验语法的合法性,如检验不对称的代码块括号、结构体标签格式错误、无效代码等;最后语法检验合法性后,采用基于抽象语法树分析法和内部调用关系分析法(包括包依赖关系分析法和函数依赖关系分析法)对智能合约源代码进行静态结构分析,获得智能合约抽象语法树、包依赖关系图和函数调用关系图等静态结构。其中智能合约抽象语法树分析法、包依赖关系分析法和函数依赖关系分析法具体实施过程如下所示。

[0040] (1)智能合约抽象语法树解析采用基于合约编译器的词法分析与语法分析,完成对合约源代码语法树的解析。其中词法分析是对源代码的字符串序列解析,并转换成词语序列(简称Token序列)。语法分析基于语法库并根据合约语法规则定义,将Token序列转化为抽象语法树结构AST(Abstract Syntax Tree,AST)。该智抽象语法树结构以整个文件为根节点,自上而下描述文件中不同层级的语法结构,如包声明、顶层变量及函数定义、包依

赖声明等。如图3所示，检测目标以Golang语言所开发的Fabric合约为例说明，其中Golang语言采用自下而上的输入流方式来构造抽象语法树结构AST，即从子树构造起，逐步向上合并，组装成一颗完整的树。抽象语法树中每个节点都有其详细的结构体声明与定义，分别代表源代码位置、与其他结构的关系等。

[0041] (2)包依赖分析解析采用基于中间码解析获得合约源代码所引入的包间依赖关系。通常为了适应多种平台环境，智能合约会采用跨平台通用语言(如Golang语言)来编写，在语言编译过程都具有中间码生成过程，使得通用语言编写合约能够运行在不同的机器中。中间码生成过程即由抽象语法树翻译到中间码的过程，基于通用语言中间码的静态单赋值特性，使得生成的包依赖逻辑关系清晰可查。

[0042] 如图4所示，整个分析过程从智能合约源程序开始，源程序视为最顶层的包，自上而下依次递归地读取依赖包信息，构建链码的包依赖关系，其中，level为当前依赖层数，maxLevel为设置的依赖最大层数。根据统计发现，自智能合约源程序开始向下读取依赖时，通常在第3层即会读取到通用语言的标准库，之后则读取到通用语言环境中与编译或计算相关的底层库依赖。因此在构建智能合约包依赖关系时，分析包括智能合约源代码开始的3层包依赖关系，即maxLevel设置的依赖最大层数为3。如图5具体实现过程如下：

[0043] a.加载智能合约源代码，设置level当前依赖层数为0，maxLevel依赖最大层数为3，记录为智能合约依赖包的根节点；

[0044] b.提取通用语言中间码中的引用import，设置level当前依赖层数为1，并记录当前根节点的子节点N

[0045] c.依次递归地对各个包引用import解析依赖，判断level是否小于maxLevel，若小于maxLevel值则将level当前依赖层数自加1，并记录当前子节点的子节点N’，并继续执行c；若大于maxLevel值则执行结束，获得最终的包依赖关系图。

[0046] (3)函数依赖分析主要借助源代码的中间码的静态单赋值特性，根据基于包含的指针分析构建函数调用关系，并对生成的函数调用关系进行筛选，得到智能合约中函数调用关系图。原始函数调用图中包含大量包括底层库在内的调用逻辑，为了提取图中对后续安全检测有用的调用关系，如图6所示，首先基于通用语言的指针库(如Golang语言的pointer库)构建最原始基于中间码的函数调用图Callgraph；其次采用深度优先的方式遍历原始调用图Callgraph中每条调用边，按照对后续安全检测有用规则(该规则可为如排除shim、peer等智能合约固定引入包相关的调用关系和排除底层库调用的逻辑等，关注合约内定义和使用的函数)，判断每个调用边上的调用函数Caller与被调用函数Callee是否满足规则，若调用边符合规则，则将该调用边上的调用函数Caller与被调用函数Callee添加到函数调用关系图集合中，依次重复执行构造出智能合约内部全部的函数调用关系图。

[0047] 根据本发明的一个实施例，核心检测引擎模块用于进行包检测、指令检测和逻辑检测及特征库自动化匹配，如图7所示，该模型首先把每个合约安全风险项抽象为静态结构特征形成特征库，包括非确定性特征库(如随机数生成、获取系统时间、访问文件系统或执行命令、引入第三方库、定义全局变量、合约结构体中声明变量、程序并发和遍历Map结构等)、数据隐私安全特征库(如跨链合约调用、未利用隐私数据机制、和未对敏感数据加密等)和逻辑安全特征库(如不可重复执行的数据查询和读写不一致等)；其次采用包检测、指令检测和逻辑检测将预处理程序模块所得到的源代码静态结构与特征库检测匹配；最后根

据匹配情况得到合约合法性检测结果。其中包检测、指令检测和逻辑检测具体实施过程如下所示。

[0048] (1)包检测首先通过深度遍历算法对包依赖关系图进行检测搜索，判定引用import关键字所定义的标准库或外部包是否在特征库的黑名单、建议库和忽略库内。其中黑名单包括标准库黑名单(如crypto/rand、math/rand、time.Date，time.Now，os/exec、os、net/http等标准库)和接入外部库黑名单(如非标准库且引用路径不以hyperledger或golang开头的外包库)，建议库包括如crypo/md5，crypo/des等，忽略库包括如以github.com/hyperledger为前缀的库。其次，对于检测到黑名单中的包引用，还需判断包引用所处依赖的层数。若该包引用处在包依赖关系图的第三依赖层，则认为该包引用与智能合约的逻辑关联性已经不大可直接忽略；若该包引用处在包依赖关系图的第一、二依赖层，则判断存在安全风险并标注；若检测到包引用处在包依赖的第一层且未使用建议库中的包，则给出意见询问是否需要加密让数据更加安全。最后，收集整理检测结果反馈给可视化检测结果展示模块。该包检测主要完成对随机数生成、获取系统时间、访问文件系统或执行命令、引入第三方库和未对敏感数据加密等安全风险项的检测。

[0049] (2)指令检测采用首先通过树深度遍历算法对抽象语法树进行检测搜索，检测抽象语法树节点特征是否与特征库匹配来判定风险项及定位风险项；其次，对与检测到抽象语法树节点特征与特征库匹配，则该节点对应的源代码存在安全风险并标注，如通过读取全局变量、智能合约结构体中声明变量、遍历Map结构、并发程序等在语法树中变量、结构体、range语句以及节点信息特征，判断所属风险及定位风险位置；通过读取对链调用、隐私数据存取等函数节点的信息，判断所属风险及定位风险位置。最后，收集整理检测结果反馈给可视化检测结果展示模块。该指令检测主要完成对定义全局变量、合约结构体中声明变量、遍历Map结构、程序并发、跨链合约调用和未利用隐私数据机制等安全风险项的检测。

[0050] (3)逻辑检测采用首先通过深度遍历算法对函数调用关系图进行检测搜索函数调用路径特征是否与特征库匹配来判定风险项及定位风险项；其次，对与检测到函数调用路径特征与特征库匹配，则该节点对应的源代码存在安全风险并标注，如不可重复执行的数据查询检测需要在函数调用关系图中，确认是否存在由反射函数Invoke到数据范围查询函数的调用路径。若路径存在，则判定存在该风险项及定位风险项；读写不一致风险也同样检测函数调用关系图中，对数据状态的读取以及写入操作生成“反向调用关系树”，树结构中存在相同父节点，判定存在该风险项及定位风险项。最后，收集整理检测结果反馈给可视化检测结果展示模块。该逻辑检测主要完成对不可重复执行的数据查询和读写不一致等安全风险项的检测。

[0051] 根据本发明的一个实施例，可视化检测结果展示模型模块基于分布式WEB工作流进行可视化展示，如图8所示，该模型主要是将合约核心检测引擎模块获得的检测结果，通过分布式WEB工作流程的模式，生成可下载至本地的合约检测报告并将检测结果可视化，可在线查看有风险项的源代码文件以及高亮显示的风险代码行，提供最直观的检测结果展示。

[0052] 首先通过分布式WEB节点组成基于区块链的分布式展示网络，将从合约核心检测引擎模块获得的检测结果通过共识机制存储在分布式展示网络账本共享；其次，通过编写智能合约(可被分布式WEB节点调用执行的运行在区块链上的代码)实现分布式的下载至本

地的合约检测报告并将检测结果可视化。最后,将展示结果的所有操作记录存储在分布式账本上,保证可审计追溯。

[0053] 尽管上面对本发明说明性的具体实施方式进行了描述,以便于本技术领域的技术人员理解本发明,且应该清楚,本发明不限于具体实施方式的范围,对本技术领域的普通技术人员来讲,只要各种变化在所附的权利要求限定和确定的本发明的精神和范围内,这些变化是显而易见的,一切利用本发明构思的发明创造均在保护之列。

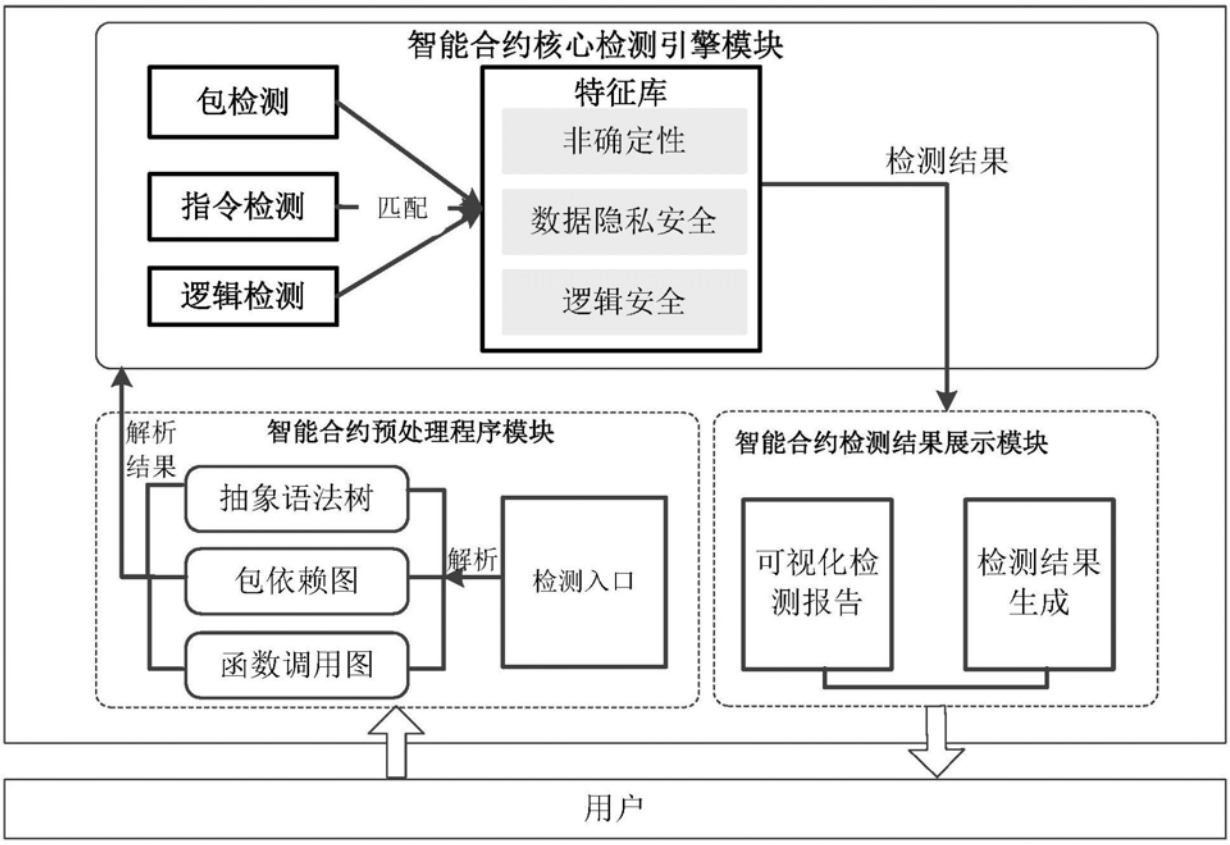

智能合约核心检测引擎模块
包检测
指令检测
逻辑检测
匹配
特征库
非确定性
数据隐私安全
逻辑安全
检测结果
解析结果
智能合约预处理程序模块
抽象语法树
包依赖图
函数调用图
解析
检测入口
智能合约检测结果展示模块
可视化检测报告
检测结果生成
用户

图1

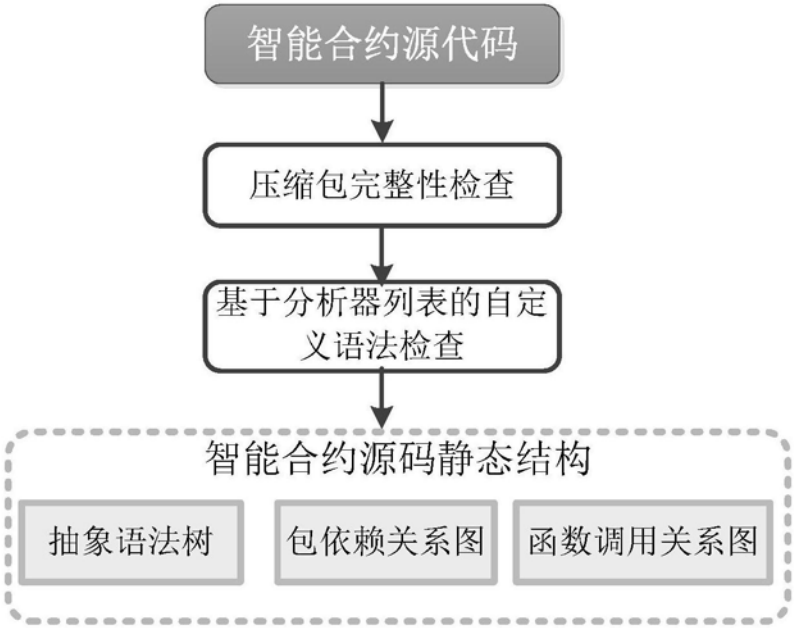

智能合约源代码
压缩包完整性检查
基于分析器列表的自定义语法检查
智能合约源码静态结构
抽象语法树
包依赖关系图
函数调用关系图

图2

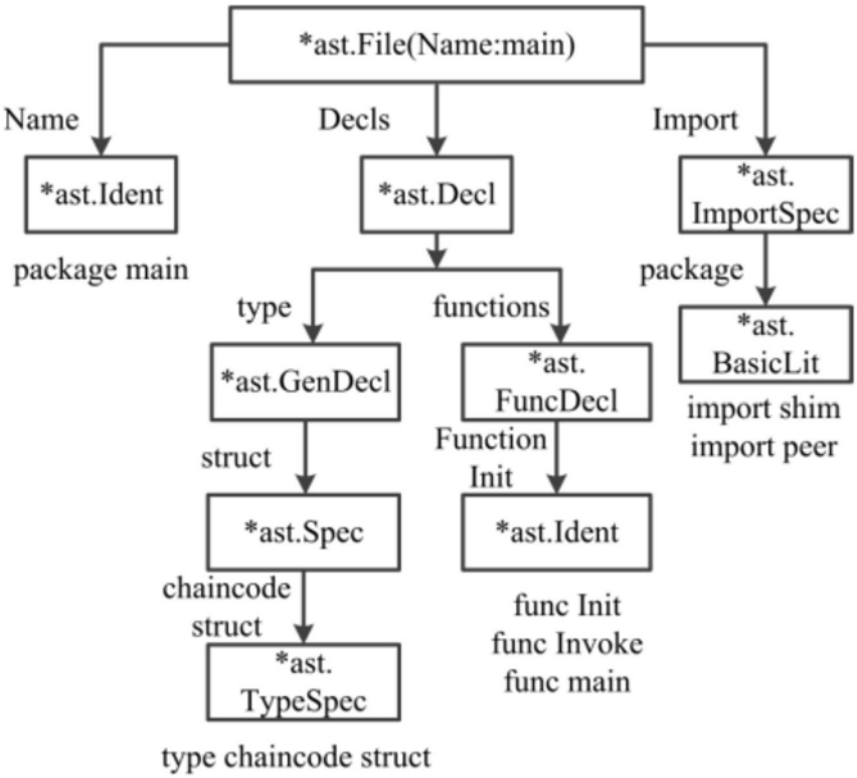

*ast.File(Name:main)
Name
Decls
Import
*ast.Ident
*ast.Decl
*ast.
ImportSpec
package main
package
type
functions
*ast.GenDecl
*ast.
FuncDecl
*ast.
BasicLit
import shim
import peer
struct
Function
Init
*ast.Spec
*ast.Ident
chaincode
struct
func Init
func Invoke
func main
*ast.
TypeSpec
type chaincode struct

图3

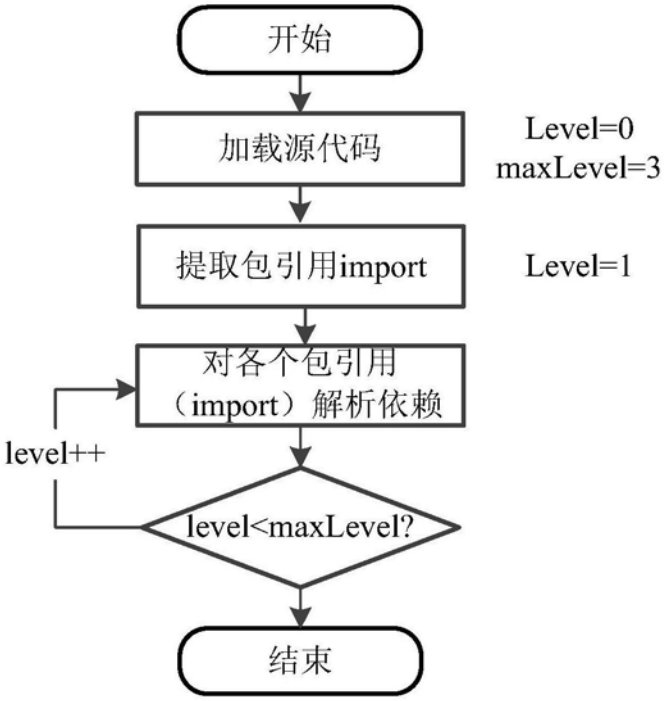

开始
加载源代码
Level=0
maxLevel=3
提取包引用import
Level=1
对各个包引用
（import）解析依赖
level++
level<maxLevel?
结束

图4

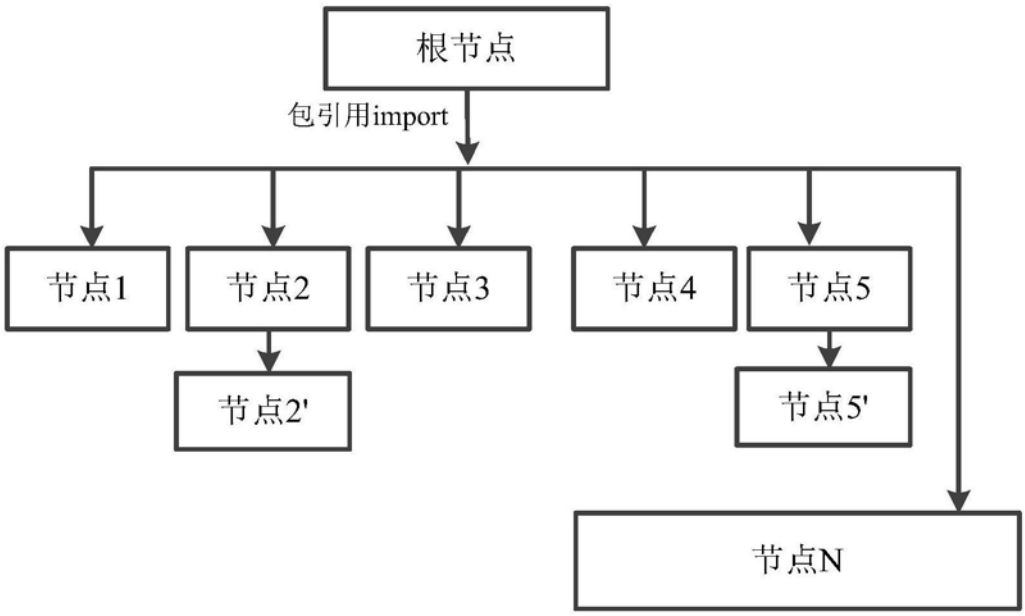

根节点
包引用import
节点1
节点2
节点3
节点4
节点5
节点2'
节点5'
节点N

图5

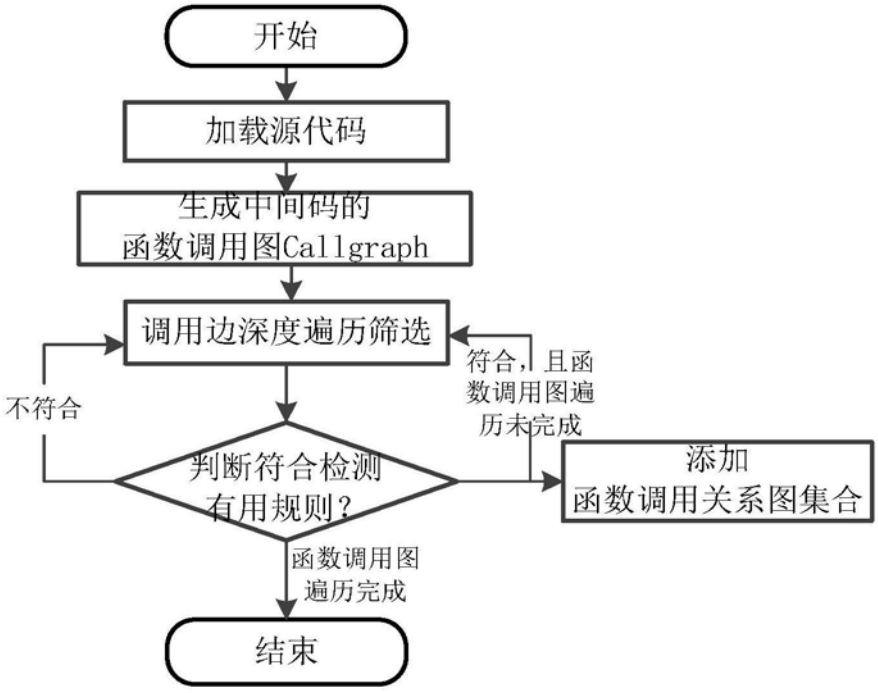

开始
加载源代码
生成中间码的
函数调用图Callgraph
调用边深度遍历筛选
符合，且函
数调用图遍
历未完成
不符合
判断符合检测
有用规则？
添加
函数调用关系图集合
函数调用图
遍历完成
结束

图6

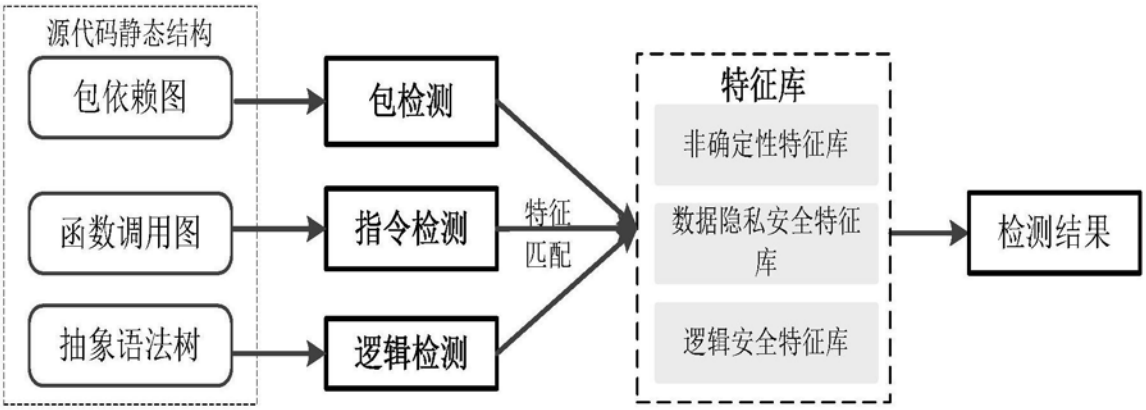

源代码静态结构
包依赖图
函数调用图
抽象语法树
包检测
指令检测
逻辑检测
特征匹配
特征库
非确定性特征库
数据隐私安全特征库
逻辑安全特征库
检测结果

图7

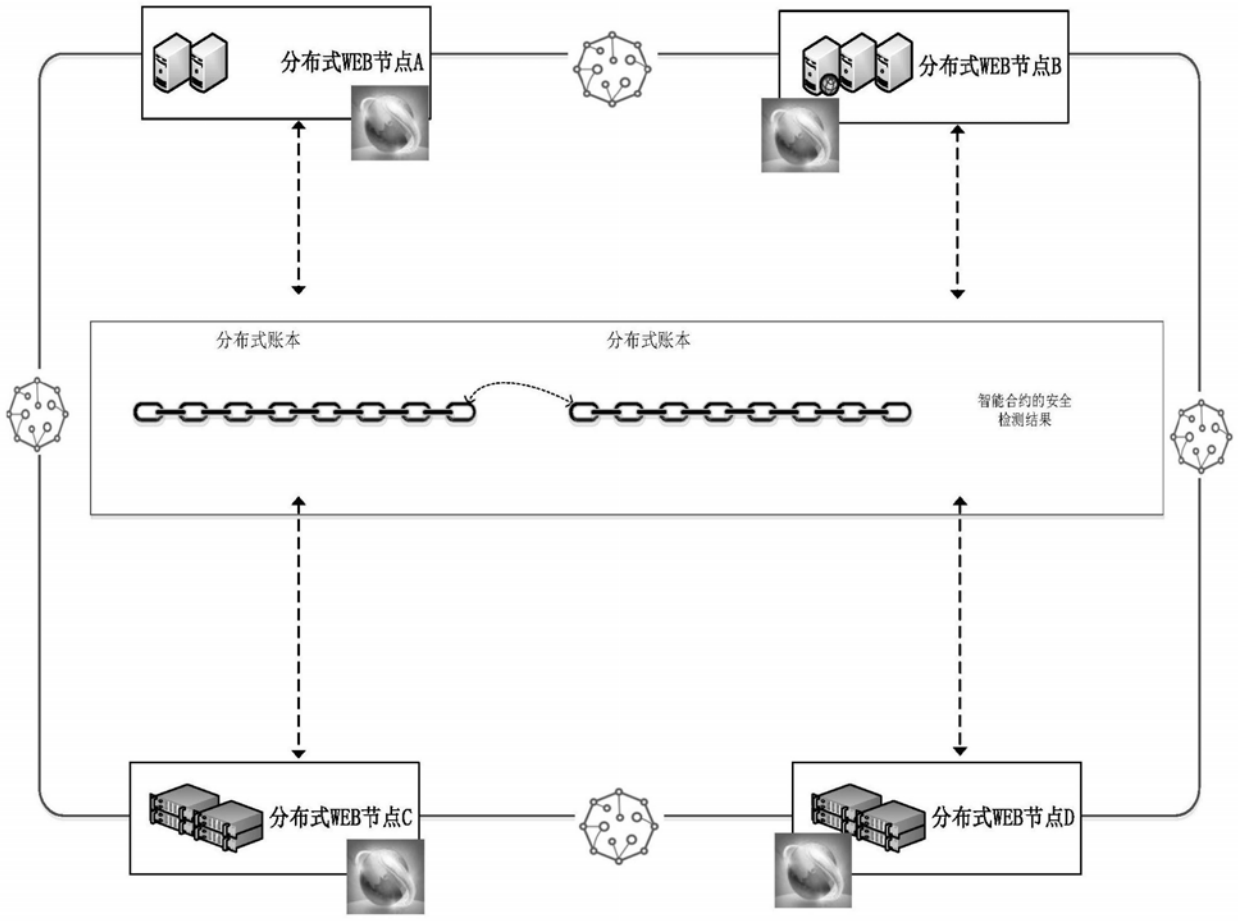

分布式WEB节点A
分布式WEB节点B
分布式账本
分布式账本
智能合约的安全检测结果
分布式WEB节点C
分布式WEB节点D

图8