(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 106295346 A
(43)申请公布日 2017.01.04

(21)申请号 201510259906.X

(22)申请日 2015.05.20

(71)申请人 深圳市腾讯计算机系统有限公司
地址 518000 广东省深圳市南山区高新区高新南一路飞亚达大厦5－10楼

(72)发明人 陈薇婷

(74)专利代理机构 深圳市深佳知识产权代理事务所（普通合伙） 44285
代理人 王仲凯

(51)Int.Cl.
G06F 21/57(2013.01)

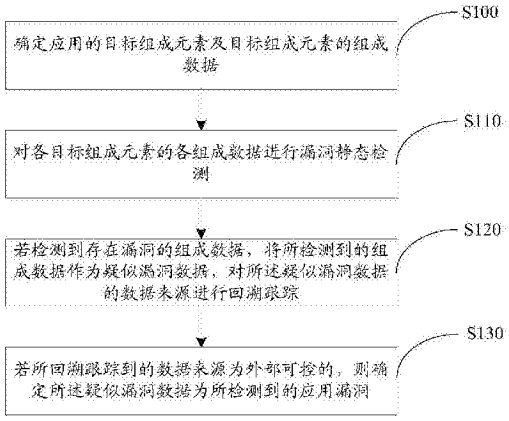
权利要求书3页 说明书11页 附图4页

(54)发明名称

一种应用漏洞检测方法、装置及计算设备

(57)摘要

本发明实施例提供一种应用漏洞检测方法、装置及计算设备，其中方法包括：确定应用的目标组成元素及目标组成元素的组成数据；对各目标组成元素的各组成数据进行漏洞静态检测；若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪；若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞。本发明可提升应用漏洞检测结果的准确性，减少了误报情况的发生。



CN 106295346 A

1. 一种应用漏洞检测方法，其特征在于，包括：

确定应用的目标组成元素及目标组成元素的组成数据；

对各目标组成元素的各组成数据进行漏洞静态检测；

若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪；

若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞。

2. 根据权利要求 1 所述的应用漏洞检测方法，其特征在于，所述目标组成元素为类，所述组成数据包括类的方法数据；

所述对各目标组成元素的各组成数据进行漏洞静态检测包括：

对各类的各方法数据中的各函数调用点进行漏洞静态检测；

所述若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据包括：

若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点；

所述对所述疑似漏洞数据的数据来源进行回溯跟踪包括：

根据函数逻辑对所述疑似函数调用点的数据来源进行回溯跟踪；

所述若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞包括：

若回溯跟踪到所述疑似函数调用点的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点。

3. 根据权来要求 2 所述的应用漏洞检测方法，其特征在于，所述对各类的各方法数据中的各函数调用点进行漏洞静态检测包括：

调用预设的漏洞规则文件，所述漏洞规则文件注册有危险函数的特征；

将各类的各方法数据中的各函数调用点对应的函数的特征，与所述漏洞规则文件中注册的危险函数的特征进行匹配处理；

若存在函数调用点对应的函数的特征，与所述漏洞规则文件中注册的危险函数的特征相匹配，则确定所述函数调用点为疑似存在漏洞的疑似函数调用点。

4. 根据权利要求 3 所述的应用漏洞检测方法，其特征在于，所述根据函数逻辑对所述疑似函数调用点的数据来源进行回溯跟踪包括：

提取所述疑似函数调用点对应函数的关键参数；

根据函数逻辑，模拟应用在真实环境中的数据流向，以回溯跟踪所述关键参数的数据来源；

所述回溯跟踪到所述疑似函数调用点的数据来源关联至外部输入包括：

回溯跟踪到所述关键参数的数据来源关联至外部输入。

5. 根据权利要求 4 所述的应用漏洞检测方法，其特征在于，所述确定应用的目标组成元素及目标组成元素的组成数据包括：

确定应用的类的树结构，所述树结构的一个节点对应一个类，一个节点下的子节点对应有类的方法数据。

6. 根据权利要求 5 所述的应用漏洞检测方法，其特征在于，还包括：构建各类所对应的符号表，所述符号表包括：各类对应的类摘要表、类中每个方法对应的函数摘要表，及函数

摘要表对应的变量信息表；

所述根据函数逻辑，模拟应用在真实环境中的数据来源向，以回溯跟踪所述关键参数的数据来源包括：

基于所述符号表回溯跟踪所述关键参数的数据来源。

7. 根据权利要求 6 所述的应用漏洞检测方法，其特征在于，所述基于所述符号表回溯跟踪所述关键参数的数据来源包括：

基于所述符号表，回溯跟踪到的数据来源为外部输入变量，则确定回溯跟踪到所述关键参数的数据来源关联至外部输入；

或，基于所述符号表，回溯跟踪到的数据来源为变量，且变量类型为基本非字符串类型，则确定回溯跟踪到所述关键参数的数据来源关联至外部输入；

或，基于所述符号表，回溯跟踪到的数据来源为变量，但所述变量不为外部输入变量，且类型不为非字符串类型时，则从所述符号表查找所述变量最近的关联值；若未查找到关联值，且所述变量不为类成员变量，则确定回溯跟踪到所述关键参数的数据来源关联至外部输入；

或，基于所述符号表，回溯跟踪到的数据来源为变量，但所述变量不为外部输入变量，且类型不为非字符串类型时，则从所述符号表查找所述变量最近的关联值；若未查找到关联值，且所述变量为类成员变量，则判断所述类成员变量是否被关联为漏洞标记，若是，则确定回溯跟踪到所述关键参数的数据来源关联至外部输入。

8. 根据权利要求 7 所述的应用漏洞检测方法，其特征在于，还包括：

基于所述符号表，回溯跟踪到的数据来源为变量，且变量类型为基本非字符串类型，则确定所述关键参数的数据来源未关联至外部输入；

或，基于所述符号表，回溯跟踪到的数据来源为变量，但所述变量不为外部输入变量，且类型不为非字符串类型时，从所述符号表查找所述变量最近的关联值；若查找到的关联值为赋值，则追踪右值对象；若查找到的关联值对应函数调用，则通过所述符号表获取所述函数摘要，并继续追踪；若未查找到关联值，且所述变量为类成员变量，则追踪所述类成员变量；

或，在追踪到函数调用时，通过所述符号表获取对应的函数摘要，追踪所获取的函数摘要所关联的参数表达式；

或，在追踪到常量时，确定所述关键参数未关联至外部输入；

或，在追踪到二元操作时，追踪对应的操作数表达式；

或，在追踪到其他类型的参数时，追踪所述其他类型的参数对应的表达式。

9. 根据权利要求 5 所述的应用漏洞检测方法，其特征在于，所述确定应用的类的树结构包括：

获取编译所述应用所采用的组成元素；

根据预先设定的抽象语法树 AST 的节点位置对应的元素类型，将各所述组成元素填入对应的 AST 节点中，得到所述应用对应的 AST；

读取所述 AST 中类的树结构。

10. 根据权利要求 9 所述的应用漏洞检测方法，其特征在于，所述获取编译所述应用所采用的组成元素包括：

对应用的安装文件进行反编译；

分析反编译后的结果，得到编译所述应用所采用的组成元素。

11. 根据权利要求 9 或 10 所述的应用漏洞检测方法，其特征在于，所述根据函数逻辑，模拟应用在真实环境中的数据来源，以回溯跟踪所述关键参数的数据来源包括：

确定所述疑似函数调用点在所述 AST 的整体树结构中的位置；

以所确定的位置为起始位置，确定所述 AST 的整体树结构中该起始位置可回溯跟踪的路径；

以所确定的路径，对所述疑似函数调用点的关键参数的数据来源进行回溯跟踪。

12. 一种应用漏洞检测装置，其特征在于，包括：

目标元素数据确定模块，用于确定应用的目标组成元素及目标组成元素的组成数据；

静态检测模块，用于对各目标组成元素的各组成数据进行漏洞静态检测；

回溯跟踪模块，用于若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪；

漏洞确定模块，用于若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞。

13. 根据权利要求 12 所述的应用漏洞检测装置，其特征在于，所述目标组成元素为类，所述组成数据包括类的方法数据；

所述静态检测模块包括：

函数静态检测单元，用于对各类的各方法数据中的各函数调用点进行漏洞静态检测；

所述回溯跟踪模块包括：

函数回溯跟踪单元，用于若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点，根据函数逻辑对所述疑似函数调用点的数据来源进行回溯跟踪；

所述漏洞确定模块包括：

漏洞函数确定单元，用于若回溯跟踪到所述疑似函数调用点的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点。

14. 根据权利要求 13 所述的应用漏洞检测装置，其特征在于，所述函数回溯跟踪单元包括：

提取子单元，用于提取所述疑似函数调用点对应函数的关键参数；

回溯跟踪执行子单元，用于根据函数逻辑，模拟应用在真实环境中的数据流向，以回溯跟踪所述关键参数的数据来源。

15. 一种计算设备，其特征在于，包括权利要求 12-14 任一项所述的应用漏洞检测装置。

一种应用漏洞检测方法、装置及计算设备

技术领域

[0001] 本发明涉及数据处理技术领域，具体涉及一种应用漏洞检测方法、装置及计算设备。

背景技术

[0002] 随着Android、IOS等智能操作系统的发展，终端设备所装载的应用越来越多，应用的安全问题也愈加受到人们的关注；应用的安全问题大部分是由于应用所存在的漏洞产生，因此对应用进行漏洞检测，对于提升应用的安全性显得尤为重要。

[0003] 目前主要通过漏洞静态检测方式检测应用中存在的漏洞，即预先制定漏洞特征，对应用的内容数据（如方法数据等）进行漏洞特征的匹配，若匹配到与漏洞特征相应的内容数据，则该内容数据的位置为应用中存在漏洞的位置。

[0004] 本发明的发明人在研究过程中发现，通过漏洞静态检测方式检测应用中存在的漏洞，虽然检测效率较高，但检测结果的准确性并不理想，尤其对于漏洞特征较不突出的应用漏洞，很容易产生误报。

发明内容

[0005] 有鉴于此，本发明实施例提供一种应用漏洞检测方法、装置及计算设备，以解决现有检测应用漏洞的方式所存在的准确性较低的问题。

[0006] 为实现上述目的，本发明实施例提供如下技术方案：

[0007] 一种应用漏洞检测方法，包括：

[0008] 确定应用的目标组成元素及目标组成元素的组成数据；

[0009] 对各目标组成元素的各组成数据进行漏洞静态检测；

[0010] 若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪；

[0011] 若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞。

[0012] 本发明实施例还提供一种应用漏洞检测装置，包括：

[0013] 目标元素数据确定模块，用于确定应用的目标组成元素及目标组成元素的组成数据；

[0014] 静态检测模块，用于对各目标组成元素的各组成数据进行漏洞静态检测；

[0015] 回溯跟踪模块，用于若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪；

[0016] 漏洞确定模块，用于若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞。

[0017] 本发明实施例还提供一种计算设备，包括上述所述的应用漏洞检测装置。

[0018] 基于上述技术方案，本发明实施例在进行应用漏洞检测时，先确定应用中可进行

应用漏洞检测的目标组成元素，及目标组成元素的组成数据；从而以漏洞静态检测方式对各目标组成元素的各组成数据进行漏洞检测，当检测到存在漏洞的组成数据时，本发明实施例并不是直接以所检测的存在漏洞的组成数据作为应用漏洞，而是将所检测到的组成数据作为疑似漏洞数据，继续对所述疑似漏洞数据的数据来源进行回溯跟踪，在所回溯跟踪到的数据来源为外部可控时，确定所述疑似漏洞数据可被外部控制，确定所述疑似漏洞数据为所检测到的应用漏洞。本发明实施例提供的应用漏洞检测方法，在通过漏洞静态检测方式检测到疑似漏洞数据时，继续对所述疑似漏洞数据进行深层次的检测，即对所述疑似漏洞数据的数据来源进行回溯跟踪，只有在所回溯跟踪到的数据来源为外部可控时，才确定所述疑似漏洞数据为所检测到的应用漏洞，从而使得最终的应用漏洞检测结果的准确性较高，减少了误报情况的发生，提升了检测结果的准确性。

附图说明

[0019] 为了更清楚地说明本发明实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据提供的附图获得其他的附图。

[0020] 图 1 为本发明实施例提供的应用漏洞检测方法的流程图；

[0021] 图 2 为本发明实施例提供的应用漏洞检测方法的另一流程图；

[0022] 图 3 为本发明实施例提供的应用漏洞检测方法的再一流程图；

[0023] 图 4 为本发明实施例提供的应用漏洞检测方法的又一流程图；

[0024] 图 5 为本发明实施例提供的确定应用的类的树结构的方法流程图；

[0025] 图 6 为本发明实施例提供的回溯跟踪关键参数的数据来源的方法流程图；

[0026] 图 7 为本发明实施例提供的应用漏洞检测装置的结构框图；

[0027] 图 8 为本发明实施例提供的应用漏洞检测装置的另一结构框图；

[0028] 图 9 为本发明实施例提供的函数静态检测单元的结构框图；

[0029] 图 10 为本发明实施例提供的函数回溯跟踪单元的结构框图；

[0030] 图 11 为本发明实施例提供的计算设备的硬件结构框图。

具体实施方式

[0031] 下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0032] 图 1 为本发明实施例提供的应用漏洞检测方法的流程图，该方法可应用于具体数据处理能力的计算设备中，如可应用于手机、平板电脑、笔记本电脑等终端设备，也可应用于服务器等网络侧设备；参照图 1，该方法可以包括：

[0033] 步骤 S100、确定应用的目标组成元素及目标组成元素的组成数据；

[0034] 可选的，此处的组成元素可以指包、类、方法、逻辑单元、字符名及值等编译应用时所采用的程序元素；目标组成元素可以是从上述组成元素中所选取的一种，通过分析则可

实现应用漏洞检测的元素；如可以漏洞静态检测方式所主要检测的组成元素作为目标组成元素；目标组成元素的组成数据可以为目标组成元素的具体内容数据；

[0035] 可选的，目标组成元素可选取类，通过对类的分析可实现应用漏洞的检测，对应的，目标组成元素的组成数据可以为类中所封装的方法等内容数据。

[0036] 步骤S110、对各目标组成元素的各组成数据进行漏洞静态检测；

[0037] 可选的，本发明实施例可设置漏洞规则文件，所述漏洞规则文件中注册有应用的漏洞特征；对于各目标组成元素的各组成数据，本发明实施例可将各组成数据的内容与漏洞规则文件中注册的漏洞特征进行匹配；

[0038] 若存在与漏洞特征相匹配的组成数据，则该组成数据为通过漏洞静态检测方式初步检测出的疑似存在漏洞的数据，本发明实施例称此类数据为疑似漏洞数据。

[0039] 步骤S120、若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪；

[0040] 可选的，可根据应用逻辑，模拟所述疑似漏洞数据的调用、产生等相关逻辑，从而对所述疑似漏洞数据的数据来源进行回溯跟踪。

[0041] 步骤S130、若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞。

[0042] 可以看出，本发明实施例在进行应用漏洞检测时，先确定应用中可进行应用漏洞检测的目标组成元素，及目标组成元素的组成数据；从而以漏洞静态检测方式对各目标组成元素的各组成数据进行漏洞检测，当检测到存在漏洞的组成数据时，本发明实施例并不是直接以所检测的存在漏洞的组成数据作为应用漏洞，而是将所检测到的组成数据作为疑似漏洞数据，继续对所述疑似漏洞数据的数据来源进行回溯跟踪，在所回溯跟踪到的数据来源为外部可控时，确定所述疑似漏洞数据可被外部控制，确定所述疑似漏洞数据为所检测到的应用漏洞。本发明实施例提供的应用漏洞检测方法，在通过漏洞静态检测方式检测到疑似漏洞数据时，继续对所述疑似漏洞数据进行深层次的检测，即对所述疑似漏洞数据的数据来源进行回溯跟踪，只有在所回溯跟踪到的数据来源为外部可控时，才确定所述疑似漏洞数据为所检测到的应用漏洞，从而使得最终的应用漏洞检测结果的准确性较高，减少了误报情况的发生，提升了检测结果的准确性。

[0043] 可选的，目标组成元素可以为类，目标组成元素的组成数据可以为类的方法数据；图2示出了本发明实施例提供的应用漏洞检测方法的另一流程图，参照图2，该方法可以包括：

[0044] 步骤S200、确定应用的类及类的方法数据；

[0045] 步骤S210、对各类的各方法数据中的各函数调用点进行漏洞静态检测；

[0046] 可选的，本发明实施例可预设漏洞规则文件，所述漏洞规则文件注册有危险函数的特征；

[0047] 在进行漏洞静态检测时，可调用预设的漏洞规则文件；将各类的各方法数据中的各函数调用点对应的函数的特征，与所述漏洞规则文件中注册的危险函数的特征进行匹配处理；若存在函数调用点对应的函数的特征，与所述漏洞规则文件中注册的危险函数的特征相匹配，则可确定所述函数调用点疑似存在漏洞，可将该函数调用点作为疑似函数调用点。

[0048] 步骤 S220、若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点；

[0049] 步骤 S230、根据函数逻辑对所述疑似函数调用点的数据来源进行回溯跟踪；

[0050] 可选的，本发明实施例可根据函数逻辑，模拟应用在真实环境中的数据流向，从而回溯跟踪到所述疑似函数调用点的数据来源。

[0051] 步骤 S240、若回溯跟踪到所述疑似函数调用点的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点。

[0052] 显然，类仅为目标组成元素的一种可选形式，本发明实施例也可根据漏洞静态检测实际可检测的元素类型选取目标组成元素。

[0053] 可选的，在对所述疑似函数调用点的数据来源进行回溯跟踪时，本发明实施例可仅对所述疑似函数调用点的关键参数的数据来源进行回溯跟踪；该关键参数可以为设定的能引起函数调用点存在漏洞的参数，具体可在所述漏洞规则文件中进行注册；对应的，图 3 示出了本发明实施例提供的应用漏洞检测方法的再一流程图，参照图 3，该方法可以包括：

[0054] 步骤 S300、确定应用的类及类的方法数据；

[0055] 步骤 S310、对各类的各方法数据中的各函数调用点进行漏洞静态检测；

[0056] 步骤 S320、若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点；

[0057] 步骤 S330、提取所述疑似函数调用点对应函数的关键参数；

[0058] 可选的，关键参数可以为函数调用点对应函数的变量、表达式、成员变量等参数。

[0059] 步骤 S340、根据函数逻辑，模拟应用在真实环境中的数据流向，以回溯跟踪所述关键参数的数据来源；

[0060] 步骤 S350、若回溯跟踪到所述关键参数的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点。

[0061] 可选的，本发明实施例可通过构建类的树结构的方式，实现应用的漏洞检测；图 4 示出了本发明实施例提供的应用漏洞检测方法的又一流程图，参照图 4，该方法可以包括：

[0062] 步骤 S400、确定应用的类的树结构，所述树结构的一个节点对应一个类，一个节点下的子节点对应有类的方法数据；

[0063] 可选的，在对一个应用进行漏洞检测时，一般只能获取到该应用的安装文件，因此本发明实施例可对应用的安装文件进行反编译，对反编译后的结果进行分析，从而确定应用中的各类，及各类的各方法数据；从而以树结构的一个节点对应一个类，类节点下的子节点对应类的方法数据的方式，构建出应用的类的树结构；显然，若可获取到应用的源代码，则可直接分析应用的源代码，确定应用中的各类，及各类的方法数据，从而构建出应用的类的树结构；

[0064] 显然，在选取其他元素作为目标组成元素时，也可通过构建相应的树结构实现应用的漏洞检测；具体在构建树结构时，可以树结构的一个节点对应一个目标组成元素，目标组成元素节点下的子节点对应具体组成数据的方式，实现树结构的构建。

[0065] 步骤 S410、针对各节点，对节点对应的各方法数据中的各函数调用点进行漏洞静态检测；

[0066] 可选的，本发明实施例可调用预设的漏洞规则文件，所述漏洞规则文件注册有危

险函数的特征；对于所述树结构中的各节点，本发明实施例可将节点所对应的类中的各方法数据的各函数调用点，与漏洞规则文件中注册的危险函数的特征进行匹配；

[0067] 若存在函数调用点对应的函数的特征，与所述漏洞规则文件中注册的危险函数的特征相匹配，则确定所述函数调用点为疑似存在漏洞的疑似函数调用点。

[0068] 步骤S420、若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点；

[0069] 步骤S430、根据函数逻辑对所述疑似函数调用点的数据来源进行回溯跟踪；

[0070] 可选的，提取所述疑似函数调用点对应函数的关键参数，根据函数逻辑，模拟应用在真实环境中的数据流向，以回溯跟踪所述关键参数的数据来源，从而实现对所述疑似函数调用点的数据来源进行回溯跟踪。

[0071] 步骤S440、若回溯跟踪到所述疑似函数调用点的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点。

[0072] 可选的，在构建类的树结构时，本发明实施例可先构建应用对应的AST(Abstract Syntax Tree，抽象语法树)，再从该AST中读取类的树结构。图5示出了本发明实施例提供的确定应用的类的树结构的方法流程图，参照图5，该方法可以包括：

[0073] 步骤500、获取编译应用所采用的组成元素；

[0074] 可选的，本发明实施例可对应用的安装文件进行反编译，从而分析反编译后的结果，得到编译所述应用所采用的组成元素；

[0075] 可选的，在分析应用的安装文件的反编译后的结果时，本发明实施例可得到应用的框架结构、语法语句内存模型(Treemodel)、字符名集合等信息；其中，Treemodel为临时内存结构，包含代码的词法语法详细信息，如控制结构、运算操作等代码关键信息；字符名是应用中最小的标志单元，代表着数据在应用运行到当前节点时的名称；应用的框架结构、语法语句内存模型(Treemodel)、字符名集合等信息中包括有类、方法、字符名及值等语法元素(treeItem)。

[0076] 步骤S510、根据预先设定的AST的节点位置对应的元素类型，将各所述组成元素填入对应的AST节点中，得到所述应用对应的AST；

[0077] 可选的，本发明实施例可定义各类型的组成元素，在AST中对应的节点位置(如定义某一类型的组成元素为节点，某一类型的组成元素为节点下的子节点等)，从而建立AST的各节点位置对应的元素类型的映射关系；在得到应用的组成元素后，可确定应用的各组成元素所对应的在AST中的节点位置，从而将各组成元素填入与对应节点中，得到所述应用对应的AST；

[0078] 可选的，以组成元素包括应用的包、类、方法、逻辑单元、字符名及值等信息为例，则可定义包对应的AST节点位置，类对应的AST节点位置，方法对应的AST节点位置(如方法可定义为相应类节点下的子节点，逻辑单元可定义为相应方法节点下的子节点等)等，从而实现AST的树框架的定义；在得到应用的组成元素后，可从应用的入口开始，逐步确定各组成元素在AST中的节点位置，最终将整个应用构建出一个完整的AST；

[0079] 步骤S520、读取所述AST中类的树结构。

[0080] 在得到应用对应的AST后，可从AST中分离出类的树结构。

[0081] 进一步，AST由于包含了应用的所有组成元素，则AST的框架结构可对应应用的执

行逻辑，本发明实施例在进行模拟应用在真实环境中的数据流向，回溯跟踪所述疑似函数调用点对应函数的关键参时，可通过 AST 对应的应用执行逻辑实现数据来源的回溯跟踪；对应的，图 6 示出了本发明实施例提供的回溯跟踪关键参数的数据来源的方法流程图，参照图 6，该方法可以包括：

[0082] 步骤 S600、在确定疑似函数调用点后，确定所述疑似函数调用点在所述 AST 的整体树结构中的位置；

[0083] 步骤 S610、以所确定的位置为起始位置，确定所述 AST 的整体树结构中该起始位置可回溯跟踪的路径；

[0084] 步骤 S620、以所确定的路径，对所述疑似函数调用点的关键参数的数据来源进行回溯跟踪。

[0085] 可选的，在得到类的树结构后，本发明实施例也可通过构建类对应的符号表的方式，实现对疑似函数调用点的关键参数的数据来源的回溯跟踪；对应的，在确定应用的类的树结构后，本发明实施例可构建类所对应的符号表；

[0086] 具体的，本发明实施例可遍历 AST 中的类节点下的方法子节点，构建每个方法的符号表；再遍历方法节点下的形参子节点，获取形参类型信息；获取方法节点下的声明、赋值、函数调用等子节点，收集变量类型，构建变量赋值、函数调用值、New 值、形参值及返回结点值等，并保存到符号表结构中；

[0087] 其中所述符号表可包括：各类对应的类摘要表、类中每个方法对应函数摘要表，及函数摘要表对应的变量信息表；

[0088] 类摘要表针对 AST 中的每个类，可收集类名、包名、import 引入的类、类成员变量等基本信息；可选的，可设置 import 引入的自定义类优先于本类分析，类成员变量信息用于类方法的分析中；

[0089] 函数摘要表针对类中的每个方法，一个方法可对应一个函数摘要表，函数摘要表可包含对应方法的数据关联关系，如赋值关系、函数调用关系、变量声明、形参信息、返回语句等信息；通过函数摘要表可管理变量信息表；

[0090] 变量信息表由两个二维映射表组成，分别为变量类型表和变量值表；变量类型表可记录方法中声明变量的类型，变量值表可记录方法中变量的数据关联；构建方式可以为：从 AST 中获取方法节点，提取方法节点下子节点的形参变量信息；然后取得方法体的节点，递归地获取相应子结点，并提取关键信息；如赋值结点 SetProperty，获取变量名 a，构建右值对象 b（含所在行、所在方法、AST 节点等信息的对象），并将 <a，b> 添入变量值表；在递归分析过程中，将变量的声明信息同时添入变量类型表。

[0091] 在构建了类所对应的符号表后，本发明实施例可基于所述符号表回溯跟踪疑似函数调用点的关键参数的数据来源；如可提取疑似函数调用点对应函数的关键参数（如函数的变量、表达式、成员变量等），根据符号表中的信息迭代查找与所述关键参数最近的前驱参数；若查找到的最近前驱为赋值，则继续回溯跟踪右值；若查找到的最近前驱为函数调用点，则判断查找到的函数调用点对应的函数是否已被函数摘要表所记录，若没有，则将查找到的函数调用点对应的函数摘要记录至函数摘要表中，并获取该函数的返回值和关联参数，继续回溯跟踪所获取的关联参数；若回溯到类成员变量，前驱可能是该类的其他方法调用隐式传递（方法影响该成员变量的值），则需要回溯与该成员变量的最近关联；

[0092] 需要注意的是，若所述关键参数关联到该方法的形参，则只要调用该方法时，此形参为外部可控的，则可确定所述疑似函数调用点为应用中存在漏洞的函数调用点；并将该方法的函数特征加入漏洞规则文件所注册的危险函数中。

[0093] 具体的，在基于所述符号表，回溯跟踪疑似函数调用点的关键参数的数据来源的具体实现可如下：

[0094] 基于所述符号表，若回溯跟踪到的数据来源为外部输入变量，则确定回溯跟踪到所述关键参数的数据来源关联至外部输入；

[0095] 基于所述符号表，若回溯跟踪到的数据来源为变量，且变量类型为基本非字符串类型，则确定回溯跟踪到所述关键参数的数据来源关联至外部输入；

[0096] 基于所述符号表，回溯跟踪到的数据来源为变量，但所述变量不为外部输入变量，且类型不为非字符串类型时，则从所述符号表查找所述变量最近的关联值；若未查找到关联值，且所述变量不为类成员变量，则确定回溯跟踪到所述关键参数的数据来源关联至外部输入；

[0097] 基于所述符号表，若回溯跟踪到的数据来源为变量，但所述变量不为外部输入变量，且类型不为非字符串类型时，则从所述符号表查找所述变量最近的关联值；若未查找到关联值，且所述变量为类成员变量，则判断所述类成员变量是否被关联为漏洞标记，若是，则确定回溯跟踪到所述关键参数的数据来源关联至外部输入。

[0098] 进一步，本发明实施例基于所述符号表，若回溯跟踪到的数据来源为变量，且变量类型为基本非字符串类型，则确定所述关键参数的数据来源未关联至外部输入；

[0099] 基于所述符号表，若回溯跟踪到的数据来源为变量，但所述变量不为外部输入变量，且类型不为非字符串类型时，则从所述符号表查找所述变量最近的关联值；若查找到的关联值为赋值，则追踪右值对象；若查找到的关联值对应函数调用，则通过所述符号表获取所述函数摘要，并继续追踪；若未查找到关联值，且所述变量为类成员变量，则追踪所述类成员变量。

[0100] 具体的，在上述追踪过程中，若追踪到函数调用，本发明实施例可通过所述符号表获取对应的函数摘要，追踪所获取的函数摘要所关联的参数表达式；

[0101] 若追踪到常量，可确定所述关键参数未关联至外部输入；

[0102] 若追踪到二元操作，追踪对应的操作数表达式；

[0103] 若追踪到其他类型的参数，可追踪该参数对应的表达式。

[0104] 进一步，在涉及到函数摘要时，若摘要到返回值的关联，对符号表中收集的每个返回点，追踪返回表达式；若追踪关联到方法形参，记录返回值与形参所在位置的关联，若追踪关联到成员变量，则记录返回值与成员变量的关联；

[0105] 若摘要到参数的关联，则对方法中每个形参变量，依次追踪形参变量在方法中的数据关联；若追踪关联到与所述形参变量不同的其他参数，记录所述形参变量与所述其他参数的关联；若追踪关联到成员变量，记录所述形参变量与所述成员变量的关联；

[0106] 若摘要到类成员变量的关联，则对每个类成员变量进行追踪；若追踪到形参，则记录所述类成员变量与所述形参的关联，且继续追踪与所述类成员变量不同的其他类成员变量，记录所述形参与所述其他类成员变量的关联。

[0107] 可选的，在通过上述方法确定了应用漏洞后，本发明实施例可以存在应用漏洞的

函数调用点所对应的类型、触发点和数据流轨迹构成应用漏洞结果数据，其中触发点和轨迹中的值由函数名和行数定位在代码中位置；将应用漏洞结果数据添加到统一列表保存并输出。

[0108] 下面基于Android虚拟机，对本发明实施例提供的Android（安卓）应用的漏洞检测测流程进行说明：

[0109] S10、载入Android应用的APK(AndroidPackage，Android安装包）文件；

[0110] S11、反编译该APK文件，得到Android应用的框架结构、语法语句内存模型(treemodel)及字符名集合等信息；并将其中每个类的信息将统一转换为smali文件(Android虚拟机所能识别的语言)，smali文件存储着一个类里完整的逻辑；

[0111] S12、获取Android应用的AST和Android虚拟机语言smali源码；具体的，可将S11所得到的程序结构、treemodel、字符名等信息转换成AST；如可将Android应用分为包、类、方法、逻辑单元、字符名及值等信息，并将这些信息分别定义为AST的各节点；然后，从应用入口开始，逐步将每种语法元素(treeItem)分别翻译转换为AST的节点，填入AST，最终将整个应用构建出一个完整的AST；

[0112] 进一步，还可将AST最终输出为一个结构化XML文档，可以让后续的漏洞检测更快捷，简易；同时可保存AST与smali源码的映射关系，方便于信息准确迅速的查找；

[0113] S13、遍历AST中的import节点，优先迭代分析引入的类；

[0114] S14、遍历AST中的类节点，分析其下的成员变量子节点，取得成员变量信息；

[0115] S15、遍历AST中的类节点下的方法子节点，构建每个方法的符号表；

[0116] 具体的，可遍历方法节点下的形参子节点，获取形参类型信息；获取方法节点下的声明、赋值、函数调用等子节点，收集变量类型，构建变量赋值、函数调用值、New值、形参值及返回结点值等，并保存到符号表结构中；

[0117] S16、基于所得到的符号表进行应用漏洞检测，具体为：

[0118] S16.1、基于S15得到的符号表，循环检查每个函数调用点，先判断是否为危险函数，若是，则提取关键参数，跳至S16.2；

[0119] S16.2、追踪关键参数的表达式，根据节点类型不同追踪不同值；若节点为变量，跳到S16.3；若节点为函数调用，则获取函数摘要，追踪摘要关联的参数的表达式，跳到S16.2；若节点为常量值，则返回安全（表示函数不存在漏洞）；若节点为二元操作，分别追踪操作数的表达式，跳到S16.2；若为其他节点类型，直接追踪对应子节点的表达式，跳到S16.2；

[0120] S16.3、追踪变量；判定变量是否为外部输入变量，若是外部输入变量，返回危险，若不是外部输入变量，则判断变量的类型是否为基本非字符串类型；若变量的类型是基本非字符串类型，则返回安全，若变量的类型不是基本非字符串类型，则从符号表查找变量最近的关联值；若存在该最近的关联值，则根据关联值的不同类型作相应的不同处理，具体的，如果该最近的关联值为赋值，则跳到S16.2追踪右值对象，如果该最近的关联值为函数调用，则通过函数摘要获取该函数调用的关联并继续追踪；若不存在该最近的关联值，而且变量是类成员变量，则追踪该类成员变量，跳到S16.4；若不存在该最近的关联值，且该变量找不到任何关联和定义，则返回危险；

[0121] S16.4、追踪类成员变量，获取类成员变量最近的关联值或函数调用；若获取到关

联值，则根据关联值的不同类型作相应的不同处理；若获取到函数调用，且该函数调用影响该类成员变量，则通过函数摘要获取关联关系，并继续追踪；如果未获取到关联值也未获取到函数调用，则判断该类成员变量是否在构造函数或其他方法中被关联为危险，若是返回危险，否则，说明此次追踪最终关联到该类成员变量；

[0122] S17、在通过函数摘要获取关联关系，并追踪时，执行如下流程：

[0123] S17.1、摘要到返回值的关联，对符号表中收集的每个返回点，追踪返回表达式，跳到S16.2；若关联到方法形参，则记录返回值与形参所在位置的关联；若关联到类成员变量，则记录返回值与类成员变量的关联；

[0124] S17.2、摘要到参数的关联，则对方法中每个形参变量，依次追踪其在方法中的数据关联，对每个形参变量，按照S16.3分析；若关联到其他参数，记录此参数与其他参数的关联；若关联到类成员变量，则记录参数与类成员变量的关联；

[0125] S17.3、摘要到类成员变量的关联，对每个类成员变量，按照S16.4分析，若追踪到形参，记录该类成员变量与形参的关联；若关联到其他类成员变量，则记录该形参与其他类成员变量的关联；

[0126] S18、跳到S15继续分析类的其他方法。

[0127] 本发明提升了应用漏洞检测结果的准确性，减少了误报情况的发生。

[0128] 下面对本发明实施例提供的应用漏洞检测装置进行介绍，下文描述的应用漏洞检测装置可与上文描述的应用漏洞检测方法相互对应参照。

[0129] 图7为本发明实施例提供的应用漏洞检测装置的结构框图，该装置可应用于具体数据处理能力的计算设备中，如可应用于手机、平板电脑、笔记本电脑等终端设备，也可应用于服务器等网络侧设备；参照图7，该装置可以包括：

[0130] 目标元素数据确定模块100，用于确定应用的目标组成元素及目标组成元素的组成数据；

[0131] 静态检测模块200，用于对各目标组成元素的各组成数据进行漏洞静态检测；

[0132] 回溯跟踪模块300，用于若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪；

[0133] 漏洞确定模块400，用于若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞。

[0134] 可选的，所述目标组成元素可以为类，所述组成数据可以为类的方法数据；对应的，图8示出了本发明实施例提供的应用漏洞检测装置的另一结构框图，结合图7和图8所示，目标元素数据确定模块100可以包括：

[0135] 类数据确定单元110，用于确定应用的类及类的方法数据；

[0136] 静态检测模块200可以包括：

[0137] 函数静态检测单元210，用于对各类的各方法数据中的各函数调用点进行漏洞静态检测；

[0138] 回溯跟踪模块300可以包括：

[0139] 函数回溯跟踪单元310，用于若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点，根据函数逻辑对所述疑似函数调用点的数据来源进行回溯跟踪；

[0140] 漏洞确定模块 400 可以包括：

[0141] 漏洞函数确定单元 410，用于若回溯跟踪到所述疑似函数调用点的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点。

[0142] 可选的，图 9 示出了本发明实施例提供的函数静态检测单元 210 的一种可选结构，参照图 9，函数静态检测单元 210 可以包括：

[0143] 调用子单元 211，用于调用预设的漏洞规则文件，所述漏洞规则文件注册有危险函数的特征；

[0144] 检测执行子单元 212，用于将各类的各方法数据中的各函数调用点对应的函数的特征，与所述漏洞规则文件中注册的危险函数的特征进行匹配处理；若存在函数调用点对应的函数的特征，与所述漏洞规则文件中注册的危险函数的特征相匹配，则确定所述函数调用点为疑似存在漏洞的疑似函数调用点。

[0145] 可选的，图 10 示出了本发明实施例提供的函数回溯跟踪单元 310 的一种可选结构，参照图 10，函数回溯跟踪单元 310 可以包括：

[0146] 提取子单元 311，用于提取所述疑似函数调用点对应函数的关键参数；

[0147] 回溯跟踪执行子单元 312，用于根据函数逻辑，模拟应用在真实环境中的数据流向，以回溯跟踪所述关键参数的数据来源。

[0148] 可选的，本发明实施例还可以类的树结构的方式，实现应用漏洞检测；具体的，可通过构建类对应的符号表实现应用漏洞检测；也在通过应用的 AST 中读取类的数结构的基础上，以 AST 的整体树结构中疑似函数调用点可回溯跟踪的路径，实现应用漏洞检测；具体内容，可参照上文应用漏洞检测方法部分的描述，此处不再赘述。

[0149] 本发明实施例还提供一种计算设备，该计算设备可以包括上述所述的应用漏洞检测装置。具体的，该计算设备可以如手机、平板电脑、笔记本电脑等终端设备，也可如服务器等网络侧设备。

[0150] 在本发明实施例中，计算设备在进行应用漏洞检测时，先通过漏洞静态检测方式进行疑似漏洞数据的检测，当检测到疑似漏洞数据时，继续对所述疑似漏洞数据进行深层次的检测，即对所述疑似漏洞数据的数据来源进行回溯跟踪，只有在所回溯跟踪到的数据来源为外部可控时，才确定所述疑似漏洞数据为所检测到的应用漏洞，从而使得最终的应用漏洞检测结果的准确性较高，减少了误报情况的发生，提升了检测结果的准确性。

[0151] 图 11 示出了本发明实施例提供的计算设备的硬件结构框图，参照图 11，该计算设备可以包括：处理器 1，通信接口 2，存储器 3 和通信总线 4；

[0152] 其中处理器 1、通信接口 2、存储器 3 通过通信总线 4 完成相互间的通信；

[0153] 可选的，通信接口 2 可以为通信模块的接口，如 GSM 模块的接口；

[0154] 处理器 1，用于执行程序；

[0155] 存储器 3，用于存放程序；

[0156] 程序可以包括程序代码，所述程序代码包括计算机操作指令。

[0157] 处理器 1 可能是一个中央处理器 CPU，或者是特定集成电路 ASIC(Application Specific Integrated Circuit)，或者是被配置成实施本发明实施例的一个或多个集成电路。

[0158] 存储器 3 可能包含高速 RAM 存储器，也可能还包括非易失性存储器 (non-volatile

memory)，例如至少一个磁盘存储器。

[0159] 其中，程序可具体用于：

[0160] 确定应用的目标组成元素及目标组成元素的组成数据；

[0161] 对各目标组成元素的各组成数据进行漏洞静态检测；

[0162] 若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪；

[0163] 若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞。

[0164] 本说明书中各个实施例采用递进的方式描述，每个实施例重点说明的都是与其他实施例的不同之处，各个实施例之间相同相似部分互相参见即可。对于实施例公开的装置而言，由于其与实施例公开的方法相对应，所以描述的比较简单，相关之处参见方法部分说明即可。

[0165] 专业人员还可以进一步意识到，结合本文中所公开的实施例描述的各示例的单元及算法步骤，能够以电子硬件、计算机软件或者二者的结合来实现，为了清楚地说明硬件和软件的可互换性，在上述说明中已经按照功能一般性地描述了各示例的组成及步骤。这些功能究竟以硬件还是软件方式来执行，取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能，但是这种实现不应认为超出本发明的范围。

[0166] 结合本文中所公开的实施例描述的方法或算法的步骤可以直接用硬件、处理器执行的软件模块，或者二者的结合来实施。软件模块可以置于随机存储器(RAM)、内存、只读存储器(ROM)、电可编程ROM、电可擦除可编程ROM、寄存器、硬盘、可移动磁盘、CD-ROM、或技术领域内所公知的任意其它形式的存储介质中。

[0167] 对所公开的实施例的上述说明，使本领域专业技术人员能够实现或使用本发明。对这些实施例的多种修改对本领域的专业技术人员来说将是显而易见的，本文中所定义的一般原理可以在不脱离本发明的精神或范围的情况下，在其它实施例中实现。因此，本发明将不会被限制于本文所示的这些实施例，而是要符合与本文所公开的原理和新颖特点相一致的最宽的范围。

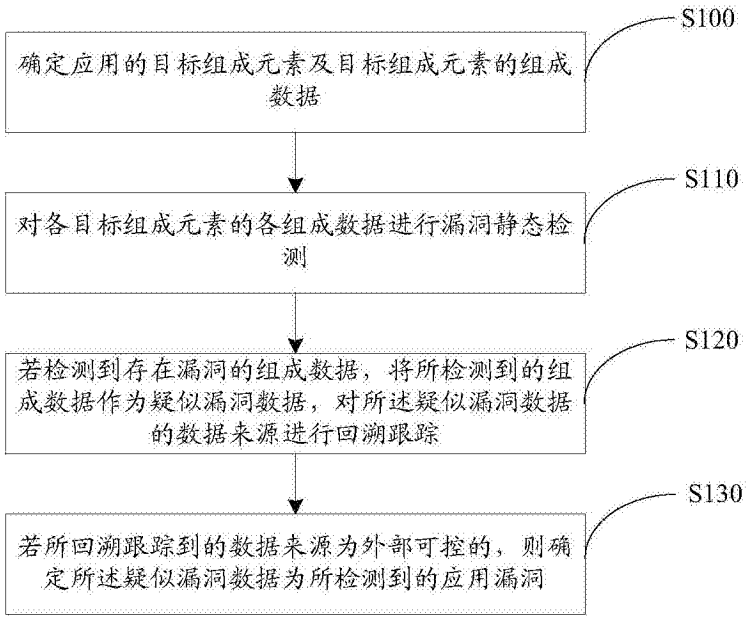

S100
确定应用的目标组成元素及目标组成元素的组成数据
S110
对各目标组成元素的各组成数据进行漏洞静态检测
S120
若检测到存在漏洞的组成数据，将所检测到的组成数据作为疑似漏洞数据，对所述疑似漏洞数据的数据来源进行回溯跟踪
S130
若所回溯跟踪到的数据来源为外部可控的，则确定所述疑似漏洞数据为所检测到的应用漏洞

图 1

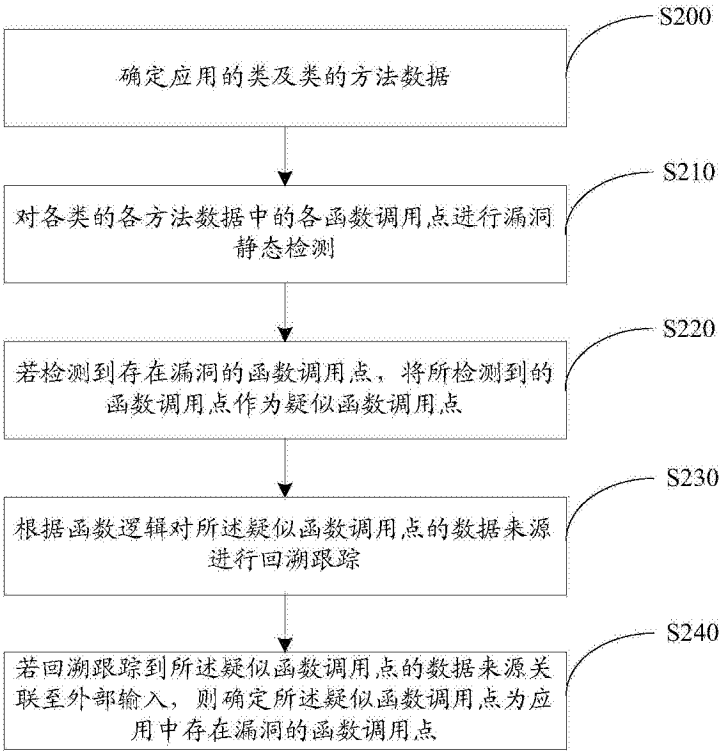

S200
确定应用的类及类的方法数据
S210
对各类的各方法数据中的各函数调用点进行漏洞静态检测
S220
若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点
S230
根据函数逻辑对所述疑似函数调用点的数据来源进行回溯跟踪
S240
若回溯跟踪到所述疑似函数调用点的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点

图 2

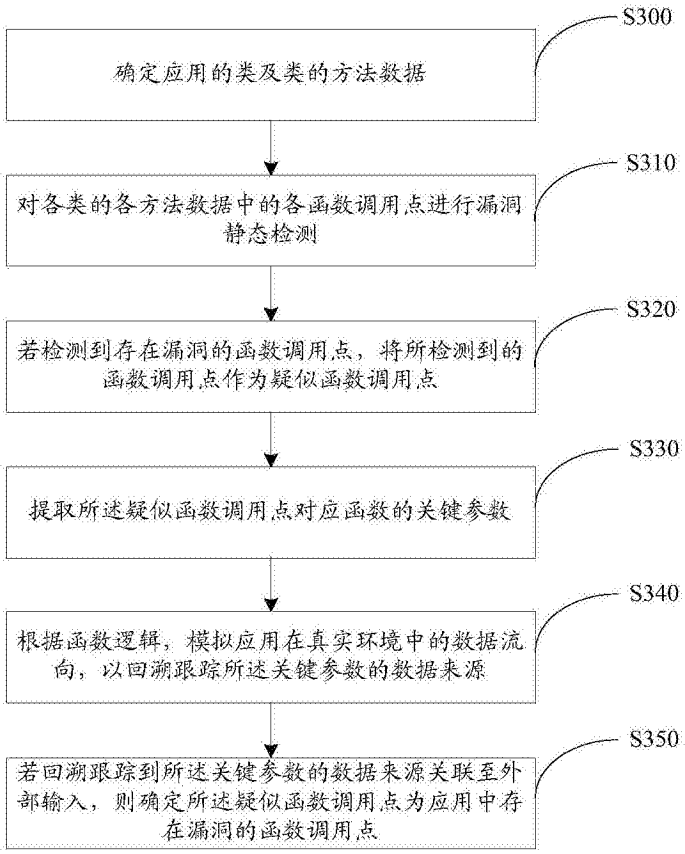

S300
确定应用的类及类的方法数据
S310
对各类的各方法数据中的各函数调用点进行漏洞静态检测
S320
若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点
S330
提取所述疑似函数调用点对应函数的关键参数
S340
根据函数逻辑，模拟应用在真实环境中的数据流向，以回溯跟踪所述关键参数的数据来源
S350
若回溯跟踪到所述关键参数的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点

图 3

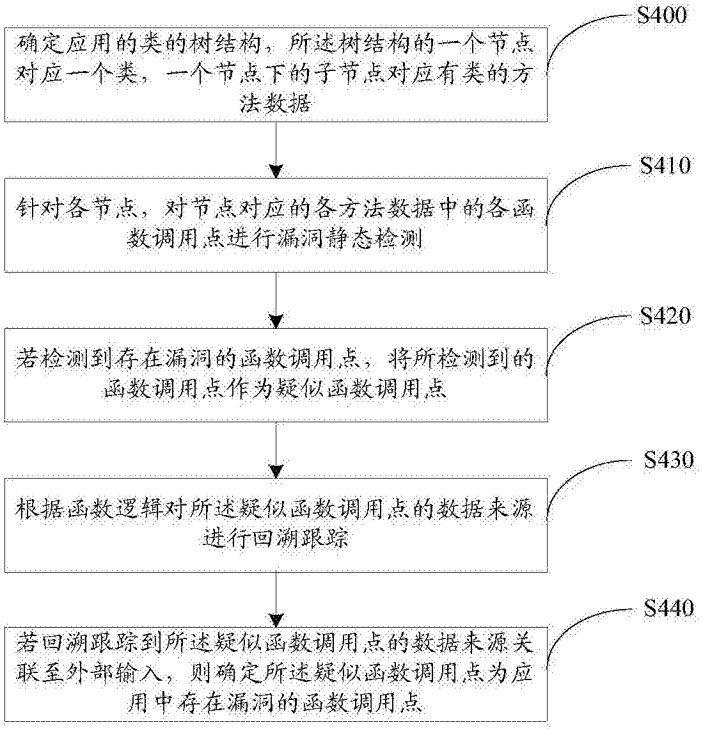

S400
确定应用的类的树结构，所述树结构的一个节点对应一个类，一个节点下的子节点对应有类的方法数据
S410
针对各节点，对节点对应的各方法数据中的各函数调用点进行漏洞静态检测
S420
若检测到存在漏洞的函数调用点，将所检测到的函数调用点作为疑似函数调用点
S430
根据函数逻辑对所述疑似函数调用点的数据来源进行回溯跟踪
S440
若回溯跟踪到所述疑似函数调用点的数据来源关联至外部输入，则确定所述疑似函数调用点为应用中存在漏洞的函数调用点

图 4

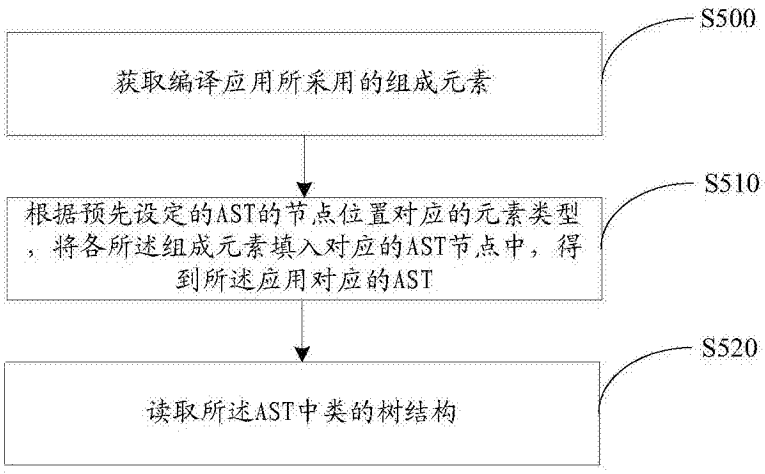

S500
获取编译应用所采用的组成元素
S510
根据预先设定的AST的节点位置对应的元素类型，将各所述组成元素填入对应的AST节点中，得到所述应用对应的AST
S520
读取所述AST中类的树结构

图 5

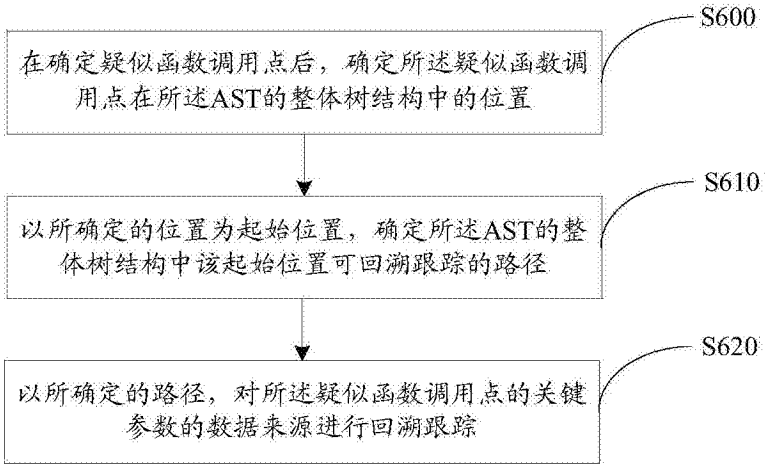

S600
在确定疑似函数调用点后，确定所述疑似函数调用点在所述AST的整体树结构中的位置
S610
以所确定的位置为起始位置，确定所述AST的整体树结构中该起始位置可回溯跟踪的路径
S620
以所确定的路径，对所述疑似函数调用点的关键参数的数据来源进行回溯跟踪

图 6

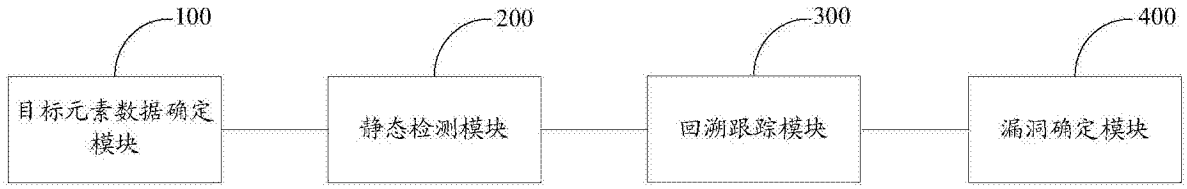

100
目标元素数据确定模块
200
静态检测模块
300
回溯跟踪模块
400
漏洞确定模块

图 7

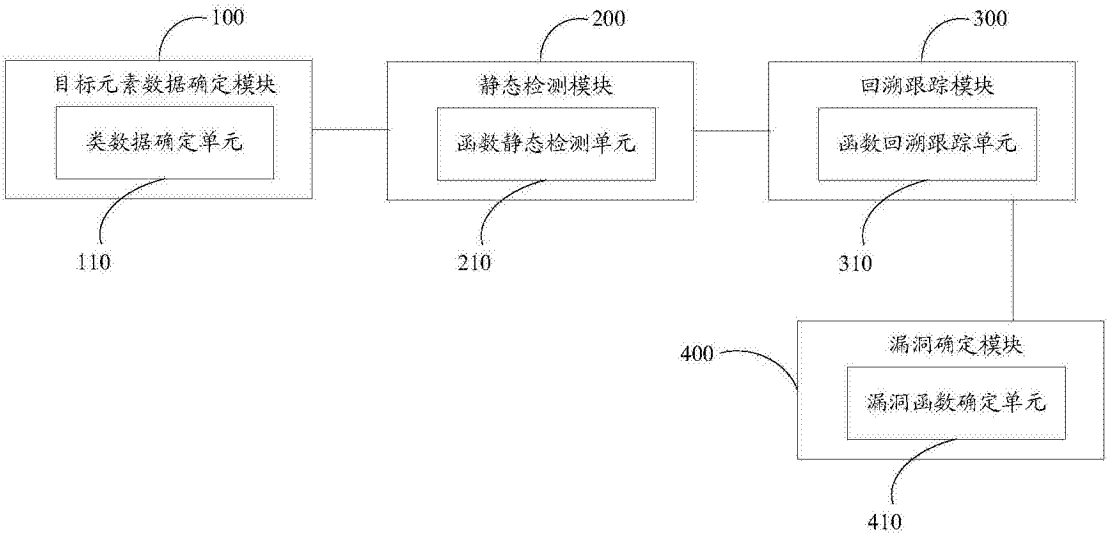

100
目标元素数据确定模块
类数据确定单元
110
200
静态检测模块
函数静态检测单元
210
300
回溯跟踪模块
函数回溯跟踪单元
310
400
漏洞确定模块
漏洞函数确定单元
410

图 8

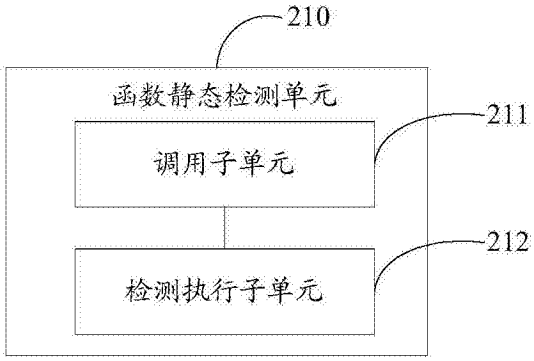

210
函数静态检测单元
211
调用子单元
212
检测执行子单元

图 9

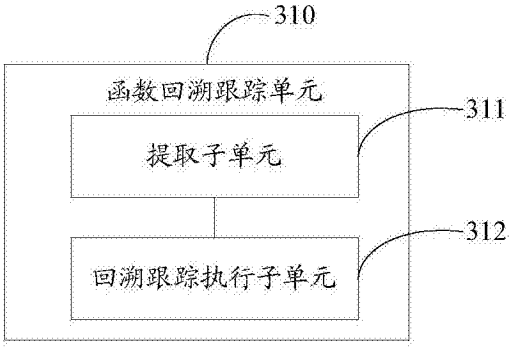

310
函数回溯跟踪单元
311
提取子单元
312
回溯跟踪执行子单元

图 10

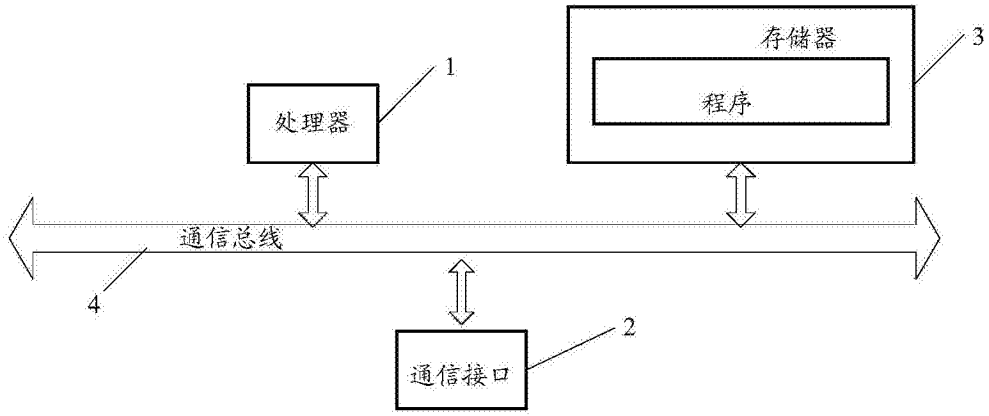

存储器
3
程序
1
处理器
通信总线
4
2
通信接口

图 11