(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 117234590 B
(45)授权公告日 2024.03.29

(21)申请号 202310966938.8

(22)申请日 2023.08.02

(65)同一申请的已公布的文献号
申请公布号 CN 117234590 A

(43)申请公布日 2023.12.15

(73)专利权人 北京握奇数据股份有限公司
地址 100102 北京市朝阳区利泽中二路2号A座6层609

(72)发明人 刘源杰 徐俊江 赵轶 郑江东 王幼君

(74)专利代理机构 北京辰权知识产权代理有限公司 11619
专利代理师 张娜

(51)Int.Cl.
G06F 9/30(2018.01)
G06F 8/41(2018.01)

(56)对比文件
CN 111736816 A,2020.10.02
CN 112434266 A,2021.03.02
CN 112631722 A,2021.04.09
CN 114428639 A,2022.05.03
CN 115033871 A,2022.09.09
CN 116069459 A,2023.05.05
CN 1172303 A,1998.02.04
US 2016335257 A1,2016.11.17
张大伟;丁文锐.Java智能卡解析优化方法.北京航空航天大学学报.2009,(第01期),全文.
李卷孺;谷大武;陆海宁.一种精简二进制代码的程序理解方法.计算机应用.2008,(第10期),全文.

审查员 徐雯晖

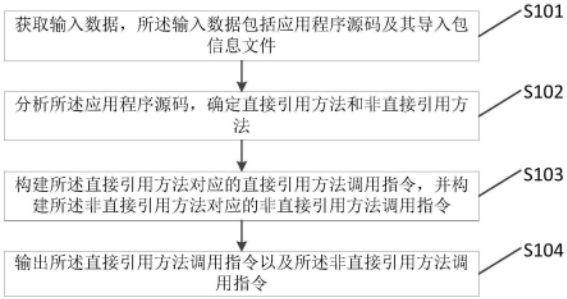
权利要求书2页 说明书11页 附图3页

(54)发明名称

指令集生成方法、装置、介质及设备

(57)摘要

本发明公开了一种指令集生成方法、装置、介质及设备,方法包括:获取输入数据,输入数据包括应用程序源码及其导入包信息文件;分析应用程序源码,确定直接引用方法和非直接引用方法;构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;输出直接引用方法调用指令以及非直接引用方法调用指令。由于本申请通过分析应用程序源码来确定直接引用方法和非直接引用方法,并提前构建直接引用方法对应的直接引用方法调用指令,避免直接引用方法在执行过程中需要解析查找对应的方法实际引用集数据,避免安全元件上的可执行文件携带大量方法相关的引用集数据,从而使得安全元件下载过程高效,兼容性好。



CN 117234590 B

1.一种指令集生成方法,其特征在于,所述方法包括:

获取输入数据,所述输入数据包括应用程序源码及其导入包信息文件;

分析所述应用程序源码,确定直接引用方法和非直接引用方法;所述直接引用方法是被调用方法在编译期可以确定且在运行期不可变的方法,所述非直接引用方法是被调用方法在编译期时无法确定且在运行期确定的方法;

构建所述直接引用方法对应的直接引用方法调用指令,并构建所述非直接引用方法对应的非直接引用方法调用指令;

输出所述直接引用方法调用指令以及所述非直接引用方法调用指令。

2.根据权利要求1所述的方法,其特征在于,所述分析所述应用程序源码,确定直接引用方法和非直接引用方法,包括:

对所述应用程序源码进行编译,得到编译中间结果;

对所述编译中间结果进行语义分析,确定所述应用程序源码中每个被调用方法的调用版本;

将存在唯一调用版本的方法识别为直接引用方法,并将不存在唯一调用版本的方法识别为非直接引用方法。

3.根据权利要求1所述的方法,其特征在于,所述构建所述直接引用方法对应的直接引用方法调用指令,包括:

获取所述应用程序源码的可执行文件;

将所述应用程序源码中每个直接引用方法的指令集数据依次放入所述可执行文件中,依次得到每个直接引用方法在所述可执行文件中的方法偏移位置;

根据预设直接方法调用类型和所述方法偏移位置,生成直接引用方法调用指令。

4.根据权利要求1所述的方法,其特征在于,所述构建所述非直接引用方法对应的非直接引用方法调用指令,包括:

将所述非直接引用方法的指令集数据依次放入可执行文件,识别所述应用程序源码对应的非直接方法结果;

根据识别的非直接方法结果以及所述导入包信息文件,生成非直接引用方法对应的方法引用;

将非直接引用方法对应的方法引用依次放入引用集,得到每个非直接引用方法的引用集数据索引;

根据预设非直接方法调用类型和每个非直接引用方法的引用集数据索引,生成非直接方法调用指令。

5.根据权利要求1所述的方法,其特征在于,所述根据预设非直接方法调用类型和每个非直接引用方法的引用集数据索引,生成非直接方法调用指令,包括:

确定预设非直接方法调用类型;

通过所述预设非直接方法调用类型,生成方法调用指令码;

通过每个非直接引用方法在预设应用集中的索引,生成非直接引用方法调用指令操作数;

将所述非直接引用方法调用指令操作数及其对应的方法调用指令码进行结合,生成非直接方法调用指令。

6.根据权利要求1所述的方法,其特征在于,所述获取输入数据之前,还包括:

通过编译器对所述应用程序源码进行编译,得到编译文件;

将所述编译文件进行打包,得到应用编程接口调用文件;

通过应用可执行文件生成器对应用程序源码或所述编译文件进行分析,获取整个包内的公开接口信息;

建立所述公开接口信息和预设链接引用之间的关联,得到关联数据;

按照预设导出信息文件格式对所述关联数据进行处理并输出,得到应用编程接口导出信息文件;

将所述应用编程接口调用文件和所述应用编程接口导出信息文件进行组合,得到导入包信息文件。

7.根据权利要求1所述的方法,其特征在于,所述获取输入数据之前,还包括:

将所述应用程序源码转换为Class文件;

根据所述Class文件以及所述导入包信息文件,解析Class信息文件;

获取所述Class信息文件的所有类信息;

根据所述类信息组织用于构成应用可执行文件的数据信息,生成应用可执行文件。

8.一种指令集生成装置,其特征在于,所述装置包括:

数据获取模块,用于获取输入数据,所述输入数据包括应用程序源码及其导入包信息文件;

源码分析模块,用于分析所述应用程序源码,确定直接引用方法和非直接引用方法;所述直接引用方法是被调用方法在编译期可以确定且在运行期不可变的方法,所述非直接引用方法是被调用方法在编译期无法确定且在运行期确定的方法;

指令构建模块,用于构建所述直接引用方法对应的直接引用方法调用指令,并构建所述非直接引用方法对应的非直接引用方法调用指令;

指令输出模块,用于输出所述直接引用方法调用指令以及所述非直接引用方法调用指令。

9.一种计算机存储介质,其特征在于,所述计算机存储介质存储有多条指令,所述指令适于由处理器加载并执行如权利要求1-7任意一项所述的方法。

10.一种设备,其特征在于,包括:处理器和存储器;其中,所述存储器存储有计算机程序,所述计算机程序适于由所述处理器加载并执行如权利要求1-7任意一项所述的方法。

指令集生成方法、装置、介质及设备

技术领域

[0001] 本发明涉及计算机技术领域,特别涉及一种指令集生成方法、装置、介质及设备。

背景技术

[0002] 安全元件可以提供身份识别、安全认证、敏感数据存储等多种安全功能。智能卡作为安全元件的一种,通过门禁卡、银行卡、公交卡等形式被广泛的应用于人们日常生活。支持虚拟机系统、可以后下载应用安全元件操作系统的出现使得安全元件应用的研发变得简单,通过安全元件中的应用,安全元件可以承载大量业务功能,提供核心安全保护,支撑各类行业的安全和业务需求。

[0003] 业务应用需要动态的管理,如下载、安装、个人化、删除等操作,这些都基于安全元件中的操作系统,而应用的运转更多依赖于安全操作系统中的虚拟机。对于安全元件中的虚拟机,依赖指令集执行引擎来完成应用逻辑和业务功能,指令集数据是通过指令集生成装置(指令集生成装置可以是编译器、解释器,也可以是对编译器输出文件进行处理的程序或软件)对源代码文件进行处理得到,方法调用指令集是指令集中必不可少的指令类型。

[0004] 对于Java虚拟机,方法调用指令集有:invokevirtual(表示虚方法调用)、invokespecial(表示特殊的实例方法调用,如实例初始化方法、私有方法和父类方法)、invokestatic(表示静态方法调用)、invokeinterface(表示接口方法调用)。对于Dalvik虚拟机,方法调用指令集有invoke-virtual(表示虚方法调用)、invoke-super(表示父类虚方法调用)、invoke-direct(表示特殊实例方法调用,如初始化方法、私有实例方法)、invoke-static(表示静态方法调用)、invoke-interface(表示接口方法调用)。

[0005] 安全元件的一个可执行文件是一个应用包数据的表现形式,其包括应用包内各种指令集数据和引用集数据(如常量池数据)。引用集数据有多种类型,对于不同类型方法有不同的方法引用数据,如虚方法、静态方法、父类方法等引用类型,这就导致安全元件上的可执行文件携带大量方法相关的引用集数据,会造成下载过程低效,甚至因安全元件容量较小而无法完成下载安装。

发明内容

[0006] 本申请实施例提供了一种指令集生成方法、装置、介质及设备。为了对披露的实施例的一些方面有一个基本的理解,下面给出了简单的概括。该概括部分不是泛泛评述,也不是要确定关键/重要组成元素或描绘这些实施例的保护范围。其唯一目的是用简单的形式呈现一些概念,以此作为后面的详细说明的序言。

[0007] 第一方面,本申请实施例提供了一种指令集生成方法,方法包括:

[0008] 获取输入数据,输入数据包括应用程序源码及其导入包信息文件;

[0009] 分析应用程序源码,确定直接引用方法和非直接引用方法;直接引用方法是被调用方法在编译期可以确定且在运行期不可变的方法,非直接引用方法是被调用方法在编译期无法确定且在运行期确定的方法;

[0010] 构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;

[0011] 输出直接引用方法调用指令以及非直接引用方法调用指令。

[0012] 可选的,分析应用程序源码,确定直接引用方法和非直接引用方法,包括:

[0013] 对应用程序源码进行编译,得到编译中间结果;

[0014] 对编译中间结果进行语义分析,确定应用程序源码中每个被调用方法的调用版本;

[0015] 将存在唯一调用版本的方法识别为直接引用方法,并将不存在唯一调用版本的方法识别为非直接引用方法。

[0016] 可选的,构建直接引用方法对应的直接引用方法调用指令,包括:

[0017] 获取应用程序源码的可执行文件;

[0018] 将应用程序源码中每个直接引用方法的指令集数据依次放入可执行文件中,依次得到每个直接引用方法在可执行文件中的方法偏移位置;

[0019] 根据预设直接方法调用类型和方法偏移位置,生成直接引用方法调用指令。

[0020] 可选的,构建非直接引用方法对应的非直接引用方法调用指令,包括:

[0021] 将非直接引用方法的指令集数据依次放入可执行文件,识别应用程序源码对应的非直接方法结果;

[0022] 根据识别的非直接方法结果以及导入包信息文件,生成非直接引用方法对应的方法引用;

[0023] 将非直接引用方法对应的方法引用依次放入引用集,得到每个非直接引用方法的引用集数据索引;

[0024] 根据预设非直接方法调用类型和每个非直接引用方法的引用集数据索引,生成非直接方法调用指令。

[0025] 可选的,根据预设非直接方法调用类型和每个非直接引用方法的引用集数据索引,生成非直接方法调用指令,包括:

[0026] 确定预设非直接方法调用类型;

[0027] 通过预设非直接方法调用类型,生成方法调用指令码;

[0028] 通过每个非直接引用方法在预设应用集中的索引,生成非直接引用方法调用指令操作数;

[0029] 将非直接引用方法调用指令操作数及其对应的方法调用指令码进行结合,生成非直接方法调用指令。

[0030] 可选的,获取输入数据之前,还包括:

[0031] 通过编译器对应用程序源码进行编译,得到编译文件;

[0032] 将编译文件进行打包,得到应用编程接口调用文件;

[0033] 通过应用可执行文件生成器对应用程序源码或编译文件进行分析,获取整个包内的公开接口信息;

[0034] 建立公开接口信息和预设链接引用之间的关联,得到关联数据;

[0035] 按照预设导出信息文件格式对关联数据进行处理并输出,得到应用编程接口导出信息文件;

[0036] 将应用编程接口调用文件和应用编程接口导出信息文件进行组合,得到导入包信息文件。

[0037] 可选的,获取输入数据之前,还包括:

[0038] 将应用程序源码转换为Class文件;

[0039] 根据Class文件以及导入包信息文件,解析Class信息文件;

[0040] 获取所述Class信息文件的所有类信息;

[0041] 根据类信息组织用于构成应用可执行文件的数据信息,生成应用可执行文件。

[0042] 第二方面,本申请实施例提供了一种指令集生成装置,装置包括:

[0043] 数据获取模块,用于获取输入数据,输入数据包括应用程序源码及其导入包信息文件;

[0044] 源码分析模块,用于分析应用程序源码,确定直接引用方法和非直接引用方法;直接引用方法是被调用方法在编译期可以确定且在运行期不可变的方法,非直接引用方法是被调用方法在编译期时无法确定且在运行期确定的方法;

[0045] 指令构建模块,用于构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;

[0046] 指令输出模块,用于输出直接引用方法调用指令以及非直接引用方法调用指令。

[0047] 第三方面,本申请实施例提供一种计算机存储介质,计算机存储介质存储有多条指令,指令适于由处理器加载并执行上述的方法步骤。

[0048] 第四方面,本申请实施例提供一种设备,可包括:处理器和存储器;其中,存储器存储有计算机程序,计算机程序适于由处理器加载并执行上述的方法步骤。

[0049] 本申请实施例提供的技术方案可以包括以下有益效果:

[0050] 在本申请实施例中,指令集生成装置首先获取输入数据,输入数据包括应用程序源码及其导入包信息文件;然后分析应用程序源码,确定直接引用方法和非直接引用方法;其次构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;最后输出直接引用方法调用指令以及非直接引用方法调用指令。由于本申请通过分析应用程序源码来确定直接引用方法和非直接引用方法,并提前构建直接引用方法对应的直接引用方法调用指令,避免直接引用方法在执行过程中需要解析查找对应的方法实际引用集数据,避免安全元件上的可执行文件携带大量方法相关的引用集数据,从而使得安全元件下载过程高效,兼容性好。

[0051] 应当理解的是,以上的一般描述和后文的细节描述仅是示例性和解释性的,并不能限制本发明。

附图说明

[0052] 此处的附图被并入说明书中并构成本说明书的一部分,示出了符合本发明的实施例,并与说明书一起用于解释本发明的原理。

[0053] 图1是本申请实施例提供的一种指令集生成方法的流程示意图;

[0054] 图2是本申请实施例提供的一种指令生成过程的过程示意框图;

[0055] 图3是本申请实施例提供的一种指令集生成装置的装置处理流程示意图;

[0056] 图4是本申请实施例提供的一种指令集生成装置的结构示意图;

[0057] 图5是本申请实施例提供的一种设备的结构示意图。

具体实施方式

[0058] 以下描述和附图充分地示出本发明的具体实施方案,以使本领域的技术人员能够实践它们。

[0059] 应当明确,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作出创造性劳动前提下所获得的所有其它实施例,都属于本发明保护的范围。

[0060] 下面的描述涉及附图时,除非另有表示,不同附图中的相同数字表示相同或相似的要素。以下示例性实施例中所描述的实施方式并不代表与本发明相一致的所有实施方式。相反,它们仅是如所附权利要求书中所详述的、本发明的一些方面相一致的装置和方法的例子。

[0061] 在本发明的描述中,需要理解的是,术语“第一”、“第二”等仅用于描述目的,而不能理解为指示或暗示相对重要性。对于本领域的普通技术人员而言,可以具体情况理解上述术语在本发明中的具体含义。此外,在本发明的描述中,除非另有说明,“多个”是指两个或两个以上。“和/或”,描述关联对象的关联关系,表示可以存在三种关系,例如,A和/或B,可以表示:单独存在A,同时存在A和B,单独存在B这三种情况。字符“/”一般表示前后关联对象是一种“或”的关系。

[0062] 本申请提供了一种指令集生成方法、装置、介质及设备,以解决上述相关技术问题中存在的问题。本申请提供的技术方案中,由于本申请通过分析应用程序源码来确定直接引用方法和非直接引用方法,并提前构建直接引用方法对应的直接引用方法调用指令,避免直接引用方法在执行过程中需要解析查找对应的方法实际引用集数据,避免安全元件上的可执行文件携带大量方法相关的引用集数据,从而使得安全元件下载过程高效,兼容性好,下面采用示例性的实施例进行详细说明。

[0063] 下面将结合附图1-附图3,对本申请实施例提供的指令集生成方法进行详细介绍。该方法可依赖于计算机程序实现,可运行于基于冯诺依曼体系的指令集生成装置上。该计算机程序可集成在应用中,也可作为独立的工具类应用运行。

[0064] 请参见图1,为本申请实施例提供了一种指令集生成方法的流程示意图。

[0065] 如图1所示,本申请实施例的方法可以包括以下步骤:

[0066] S101,获取输入数据,输入数据包括应用程序源码及其导入包信息文件;

[0067] 其中,应用程序源码是工程师针对具体的业务逻辑所编辑的代码文件,例如采用Java语言对某个业务逻辑所编辑的源代码。导入包信息中包含了一个包内所有类的公开的接口信息(如public class,public method等)。

[0068] 在本申请实施例中,在生成导入包信息文件时,首先通过编译器对应用程序源码进行编译,得到编译文件;再将编译文件进行打包,得到应用编程接口调用文件;然后通过应用可执行文件生成器对应用程序源码或编译文件进行分析,获取整个包内的公开接口信息;其次建立公开接口信息和预设链接引用之间的关联,得到关联数据;再按照预设导出信息文件格式对关联数据进行处理并输出,得到应用编程接口导出信息文件;最后将应用编程接口调用文件和应用编程接口导出信息文件进行组合,得到导入包信息文件。

[0069] 在一种可能的实现方式中,导入包信息文件包括导入包提供的应用编程接口调用文件和应用编程接口导出信息文件。

[0070] 应用编程接口调用文件生成过程:编译器对应用编程接口源码进行编译,生成中间文件(如Java中的.class文件),并将中间文件打包而成(如Java中的.jar文件)。

[0071] 应用编程接口导出信息文件生成过程:应用可执行文件生成器对应用包内的应用源码或中间文件进行分析,获取整个包内的公开接口信息,建立公开接口信息和链接引用的关联,最后按照导出信息文件格式输出。

[0072] 具体的,在生成应用可执行文件时,首先将应用程序源码转换为Class文件;然后根据Class文件以及导入包信息文件,解析Class信息文件;其次获取所述Class信息文件的所有类信息;最后根据类信息组织用于构成应用可执行文件的数据信息,生成应用可执行文件。

[0073] 在本申请实施例中,在指令集生成时,需要获取输入数据,输入数据包括应用程序源码及其导入包信息文件。

[0074] S102,分析应用程序源码,确定直接引用方法和非直接引用方法;直接引用方法是被调用方法在编译期可以确定且在运行期不可变的方法,非直接引用方法是被调用方法在编译期无法确定且在运行期确定的方法;

[0075] 在本申请实施例中,在分析应用程序源码,确定直接引用方法和非直接引用方法时,首先对应用程序源码进行编译,得到编译中间结果;然后对编译中间结果进行语义分析,确定应用程序源码中每个被调用方法的调用版本;最后将存在唯一调用版本的方法识别为直接引用方法,并将不存在唯一调用版本的方法识别为非直接引用方法。

[0076] 在一种可能的实现方式中,确定直接方法和非直接方法的过程,首先指令生成器对代码编译得到中间结果;然后对中间结果进行语义分析,分析每个被调用方法的实现版本;最后能够确定唯一调用版本的方法是直接方法,否则就是非直接方法。

[0077] S103,构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;

[0078] 在本申请实施例中,在构建直接引用方法对应的直接引用方法调用指令时,首先获取应用程序源码的可执行文件;然后将应用程序源码中每个直接引用方法的指令集数据依次放入可执行文件中,依次得到每个直接引用方法在可执行文件中的方法偏移位置;最后根据方法偏移位置,生成直接引用方法调用指令。基于寄存器的虚拟机指令的直接引用方法调用指令例如表1所示。基于栈的虚拟机指令的直接引用方法调用指令例如表2所示。

[0079] 表1

[0080]

指令格式	示例	说明
op CC_h CC_l AA BB	invoke-static-direct/r rAA BB CCCC	个数可变、编号连续的寄存器参数 rAA：8 位寄存器，存放第一个参数寄存器的编号 BB：8 位无符号常数，参数寄存器的个数 CCCC：方法偏移

[0081] 表2

[0082]

指令格式	示例	说明
op method_offset1 method_offset2	invokestatic-direct *method_offset1* *method_offset2*	method_offset1 和 method_offset2 构成方法偏移

[0083] 例如,首先将每个方法的指令集数据依次放入可执行文件;然后依次得到每个方法在可执行文件中的偏移位置;最后使用之前识别的直接方法结果,结合方法调用类型和方法在可执行文件中的偏移生成直接方法调用指令。

[0084] 在本申请实施例中,在构建非直接引用方法对应的非直接引用方法调用指令时,首先将非直接引用方法的指令集数据依次放入可执行文件,识别应用程序源码对应的非直接方法结果;然后根据识别的非直接方法结果以及导入包信息文件,生成非直接引用方法对应的方法引用;其次将非直接引用方法对应的方法引用依次放入引用集,得到每个非直接引用方法的引用集数据索引;最后根据预设非直接方法调用类型和每个非直接引用方法的引用集数据索引,生成非直接方法调用指令。基于寄存器的虚拟机指令的直接引用方法调用指令例如表3所示。基于栈的虚拟机指令的直接引用方法调用指令例如表4所示。

[0085] 表3

[0086]

指令格式	助记符	说明
op CC_h CC_l AA BB	invoke-static /r rAA BB CCCC	个数可变、编号连续的

[0087]

		寄存器参数 rAA：8 位寄存器，存放第一个参数寄存器的编号 BB：8 位无符号常数，参数寄存器的个数 CCCC：引用集数据索引

[0088] 表4

[0089]

指令格式	助记符	说明
op indexbyte1 indexbyte2	invokestatic *indexbyte1* *indexbyte2*	indexbyte1 和 indexbyte2 构成引用集数据索引

[0090] 例如,构建非直接引用方法对应的非直接引用方法调用指令过程如下:将每个方法的指令集数据依次放入可执行文件;根据之前识别的非直接方法结果,结合导入包信息文件,生成对应的方法引用,并将方法引用依次放入引用集,得到方法的引用集索引;根据非直接方法调用类型和方法的引用集索引生成非直接方法调用指令。

[0091] 具体的,在根据预设非直接方法调用类型和每个非直接引用方法的引用集数据索引,生成非直接方法调用指令时,首先确定预设非直接方法调用类型;然后通过预设非直接方法调用类型,生成方法调用指令码;其次通过每个非直接引用方法在预设应用集中的索引,生成非直接引用方法调用指令操作数;最后将非直接引用方法调用指令操作数及其对应的方法调用指令码进行结合,生成非直接方法调用指令。

[0092] 例如,通过非直接方法调用类型,生成方法调用指令码(如invokestatic);通过非直接方法在应用集中的索引(如常量池索引),生成方法调用指令操作数(如indexbyte1 indexbyte2)将两者结合,生成最终的方法调用字节码(如invokestatic indexbyte1 indexbyte2),得到非直接方法调用指令。

[0093] 具体的,在下载可执行文件文件时,通过解析和处理各种数据信息(如常量池信息、类信息、方法信息)完成方法调用指令中方法引用的链接。

[0094] S104,输出直接引用方法调用指令以及非直接引用方法调用指令。

[0095] 在一种可能的实现方式中,在得到直接引用方法调用指令以及非直接引用方法调用指令,可将其输出。

[0096] 例如图2所示,图2是本申请提供的一种指令生成过程的过程示意框图,首先获取输入数据,该输入数据包括应用源码以及导入包信息,将该输出数据输入指令集生成装置中,经过该装置处理后输出数据,该数据包括指令集以及辅助该指令集执行的其他数据,其他数据包括直接引用的引用集数据、其他引用集数据以及其他数据。

[0097] 例如图3所示,图3是本申请提供的一种指令集生成装置的装置处理流程示意图,首先指令集生成器收到输入数据,包含应用源码和导入包信息,指令集生成器分析输入数据,确定直接引用方法和非直接引用方法,然后对于直接引用方法,生成直接引用方法调用指令,其次对于非直接引用方法,生成引用数据集和指令集,生成非直接引用方法调用指令,最后将生成的指令集、引用集数据、其他数据输出。

[0098] 需要说明的是,通过剔除引用集数据减少引用集大小,从而减少应用可执行文件数据大小;通过直接引用改善方法调用指令集执行速度,从而提高应用逻辑执行效率。

[0099] 在本申请实施例中,指令集生成装置首先获取输入数据,输入数据包括应用程序源码及其导入包信息文件;然后分析应用程序源码,确定直接引用方法和非直接引用方法;其次构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;最后输出直接引用方法调用指令以及非直接引用方法调用指令。由于本申请通过分析应用程序源码来确定直接引用方法和非直接引用方法,并提前构建直接引用方法对应的直接引用方法调用指令,避免直接引用方法在执行过程中需要解析查找对应的方法实际引用集数据,避免安全元件上的可执行文件携带大量方法相关的引用集数据,从而使得安全元件下载过程高效,兼容性好。

[0100] 下述为本发明装置实施例,可以用于执行本发明方法实施例。对于本发明装置实施例中未披露的细节,请参照本发明方法实施例。

[0101] 请参见图4,其示出了本发明一个示例性实施例提供的指令集生成装置的结构示意图。该指令集生成装置可以通过软件、硬件或者两者的结合实现成为设备的全部或一部分。该装置1包括数据获取模块10、源码分析模块20、指令构建模块30、指令输出模块40。

[0102] 数据获取模块10,用于获取输入数据,输入数据包括应用程序源码及其导入包信息文件;

[0103] 源码分析模块20,用于分析应用程序源码,确定直接引用方法和非直接引用方法;直接引用方法是被调用方法在编译期可以确定且在运行期不可变的方法,非直接引用方法是被调用方法在编译期时无法确定且在运行期确定的方法;

[0104] 指令构建模块30,用于构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;

[0105] 指令输出模块40,用于输出直接引用方法调用指令以及非直接引用方法调用指令。

[0106] 需要说明的是,上述实施例提供的指令集生成装置在执行指令集生成方法时,仅以上述各功能模块的划分进行举例说明,实际应用中,可以根据需要而将上述功能分配由不同的功能模块完成,即将设备的内部结构划分成不同的功能模块,以完成以上描述的全部或者部分功能。另外,上述实施例提供的指令集生成装置与指令集生成方法实施例属于同一构思,其体现实现过程详见方法实施例,这里不再赘述。

[0107] 上述本申请实施例序号仅仅为了描述,不代表实施例的优劣。

[0108] 在本申请实施例中,指令集生成装置首先获取输入数据,输入数据包括应用程序源码及其导入包信息文件;然后分析应用程序源码,确定直接引用方法和非直接引用方法;其次构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;最后输出直接引用方法调用指令以及非直接引用方法调用指令。由

于本申请通过分析应用程序源码来确定直接引用方法和非直接引用方法,并提前构建直接引用方法对应的直接引用方法调用指令,避免直接引用方法在执行过程中需要解析查找对应的方法实际引用集数据,避免安全元件上的可执行文件携带大量方法相关的引用集数据,从而使得安全元件下载过程高效,兼容性好。

[0109] 本发明还提供一种计算机可读介质,其上存储有程序指令,该程序指令被处理器执行时实现上述各个方法实施例提供的指令集生成方法。

[0110] 本发明还提供了一种包含指令的计算机程序产品,当其在计算机上运行时,使得计算机执行上述各个方法实施例的指令集生成方法。

[0111] 请参见图5,为本申请实施例提供了一种设备的结构示意图。如图5所示,设备1000可以包括:至少一个处理器1001,至少一个网络接口1004,用户接口1003,存储器1005,至少一个通信总线1002。

[0112] 其中,通信总线1002用于实现这些组件之间的连接通信。

[0113] 其中,用户接口1003可以包括显示屏(Display)、摄像头(Camera),可选用户接口1003还可以包括标准的有线接口、无线接口。

[0114] 其中,网络接口1004可选的可以包括标准的有线接口、无线接口(如WI-FI接口)。

[0115] 其中,处理器1001可以包括一个或者多个处理核心。处理器1001利用各种接口和线路连接整个电子设备1000内的各个部分,通过运行或执行存储在存储器1005内的指令、程序、代码集或指令集,以及调用存储在存储器1005内的数据,执行电子设备1000的各种功能和处理数据。可选的,处理器1001可以采用数字信号处理(Digital Signal Processing,DSP)、现场可编程门阵列(Field-Programmable Gate Array,FPGA)、可编程逻辑阵列(Programmable Logic Array,PLA)中的至少一种硬件形式来实现。处理器1001可集成中央处理器(Central Processing Unit,CPU)、图像处理器(Graphics Processing Unit,GPU)和调制解调器等中的一种或几种的组合。其中,CPU主要处理操作系统、用户界面和应用程序等;GPU用于负责显示屏所需要显示的内容的渲染和绘制;调制解调器用于处理无线通信。可以理解的是,上述调制解调器也可以不集成到处理器1001中,单独通过一块芯片进行实现。

[0116] 其中,存储器1005可以包括随机存储器(Random Access Memory,RAM),也可以包括只读存储器(Read-Only Memory)。可选的,该存储器1005包括非瞬时性计算机可读介质(non-transitory computer-readable storage medium)。存储器1005可用于存储指令、程序、代码、代码集或指令集。存储器1005可包括存储程序区和存储数据区,其中,存储程序区可存储用于实现操作系统的指令、用于至少一个功能的指令(比如触控功能、声音播放功能、图像播放功能等)、用于实现上述各个方法实施例的指令等;存储数据区可存储上面各个方法实施例中涉及到的数据等。存储器1005可选的还可以是至少一个位于远离前述处理器1001的存储装置。如图5所示,作为一种计算机存储介质的存储器1005中可以包括操作系统、网络通信模块、用户接口模块以及指令集生成应用程序。

[0117] 在图5所示的设备1000中,用户接口1003主要用于为用户提供输入的接口,获取用户输入的数据;而处理器1001可以用于调用存储器1005中存储的指令集生成应用程序,并具体执行以下操作:

[0118] 获取输入数据,输入数据包括应用程序源码及其导入包信息文件;

[0119] 分析应用程序源码,确定直接引用方法和非直接引用方法;直接引用方法是被调用方法在编译期可以确定且在运行期不可变的方法,非直接引用方法是被调用方法在编译期无法确定且在运行期确定的方法;

[0120] 构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;

[0121] 输出直接引用方法调用指令以及非直接引用方法调用指令。

[0122] 在一个实施例中,处理器1001在执行分析应用程序源码,确定直接引用方法和非直接引用方法时,还执行以下操作:

[0123] 对应用程序源码进行编译,得到编译中间结果;

[0124] 对编译中间结果进行语义分析,确定应用程序源码中每个被调用方法的调用版本;

[0125] 将存在唯一调用版本的方法作为直接引用方法,并将不存在唯一调用版本的方法作为非直接引用方法。

[0126] 在一个实施例中,处理器1001在执行构建直接引用方法对应的直接引用方法调用指令时,具体执行以下操作:

[0127] 获取应用程序源码的可执行文件;

[0128] 将应用程序源码中每个直接引用方法的指令集数据依次放入可执行文件中,依次得到每个直接引用方法在可执行文件中的方法偏移位置;

[0129] 根据方法偏移位置,生成直接引用方法调用指令。

[0130] 在一个实施例中,处理器1001在执行构建非直接引用方法对应的非直接引用方法调用指令时,具体执行以下操作:

[0131] 将非直接引用方法的指令集数据依次放入可执行文件,识别应用程序源码对应的非直接方法结果;

[0132] 根据识别的非直接方法结果以及导入包信息文件,生成非直接引用方法对应的方法引用;

[0133] 将非直接引用方法对应的方法引用依次放入引用集,得到每个非直接引用方法的引用集数据索引;

[0134] 根据预设非直接方法调用类型和每个非直接引用方法的引用集数据索引,生成非直接方法调用指令。

[0135] 在一个实施例中,处理器1001在执行根据预设非直接方法调用类型和每个非直接引用方法的引用集数据索引,生成非直接方法调用指令时,具体执行以下操作:

[0136] 确定预设非直接方法调用类型;

[0137] 通过预设非直接方法调用类型,生成方法调用指令码;

[0138] 通过每个非直接引用方法在预设应用集中的索引,生成非直接引用方法调用指令操作数;

[0139] 将非直接引用方法调用指令操作数及其对应的方法调用指令码进行结合,生成非直接方法调用指令。

[0140] 在一个实施例中,处理器1001在执行获取输入数据之前时,还执行以下操作:

[0141] 通过编译器对应用程序源码进行编译,得到编译文件;

[0142] 将编译文件进行打包,得到应用编程接口调用文件;

[0143] 通过应用可执行文件生成器对应用程序源码或编译文件进行分析,获取整个包内的公开接口信息;

[0144] 建立公开接口信息和预设链接引用之间的关联,得到关联数据;

[0145] 按照预设导出信息文件格式对关联数据进行处理并输出,得到应用编程接口导出信息文件;

[0146] 将应用编程接口调用文件和应用编程接口导出信息文件进行组合,得到导入包信息文件。

[0147] 在一个实施例中,处理器1001在执行获取输入数据之前时,还执行以下操作:

[0148] 将应用程序源码转换为Class文件;

[0149] 根据Class文件以及导入包信息文件,解析Class信息文件;

[0150] 获取所述Class信息文件的所有类信息;

[0151] 根据类信息组织用于构成应用可执行文件的数据信息,生成应用可执行文件。

[0152] 在本申请实施例中,指令集生成装置首先获取输入数据,输入数据包括应用程序源码及其导入包信息文件;然后分析应用程序源码,确定直接引用方法和非直接引用方法;其次构建直接引用方法对应的直接引用方法调用指令,并构建非直接引用方法对应的非直接引用方法调用指令;最后输出直接引用方法调用指令以及非直接引用方法调用指令。由于本申请通过分析应用程序源码来确定直接引用方法和非直接引用方法,并提前构建直接引用方法对应的直接引用方法调用指令,避免直接引用方法在执行过程中需要解析查找对应的方法实际引用集数据,避免安全元件上的可执行文件携带大量方法相关的引用集数据,从而使得安全元件下载过程高效,兼容性好。

[0153] 本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程,是可以通过计算机程序来指令相关的硬件来完成,指令集生成的程序可存储于计算机可读取存储介质中,该程序在执行时,可包括如上述各方法的实施例的流程。其中,指令集生成的程序的存储介质可为磁碟、光盘、只读存储记忆体或随机存储记忆体等。

[0154] 以上所揭露的仅为本申请较佳实施例而已,当然不能以此来限定本申请之权利范围,因此依本申请权利要求所作的等同变化,仍属本申请所涵盖的范围。

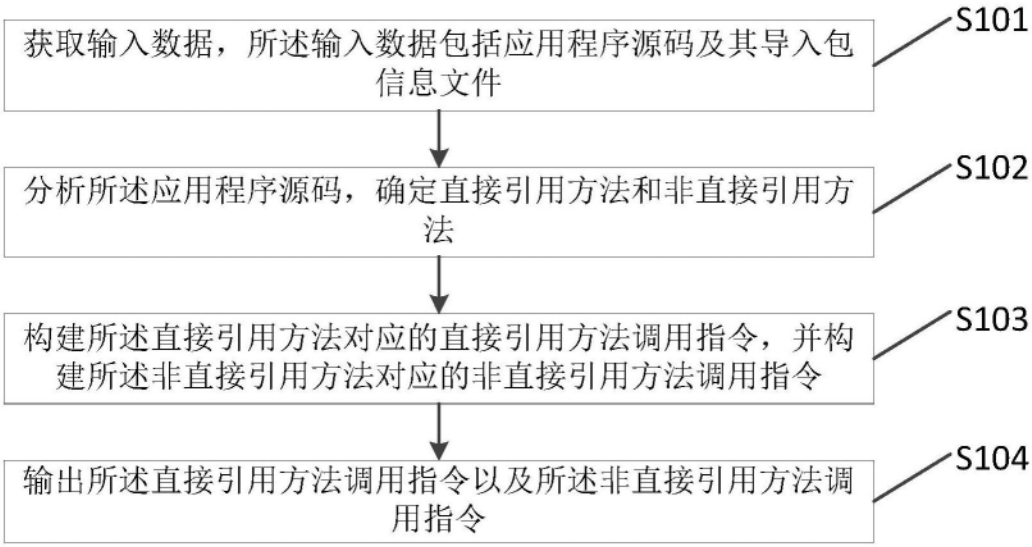

获取输入数据，所述输入数据包括应用程序源码及其导入包信息文件
S101
分析所述应用程序源码，确定直接引用方法和非直接引用方法
S102
构建所述直接引用方法对应的直接引用方法调用指令，并构建所述非直接引用方法对应的非直接引用方法调用指令
S103
输出所述直接引用方法调用指令以及所述非直接引用方法调用指令
S104

图1

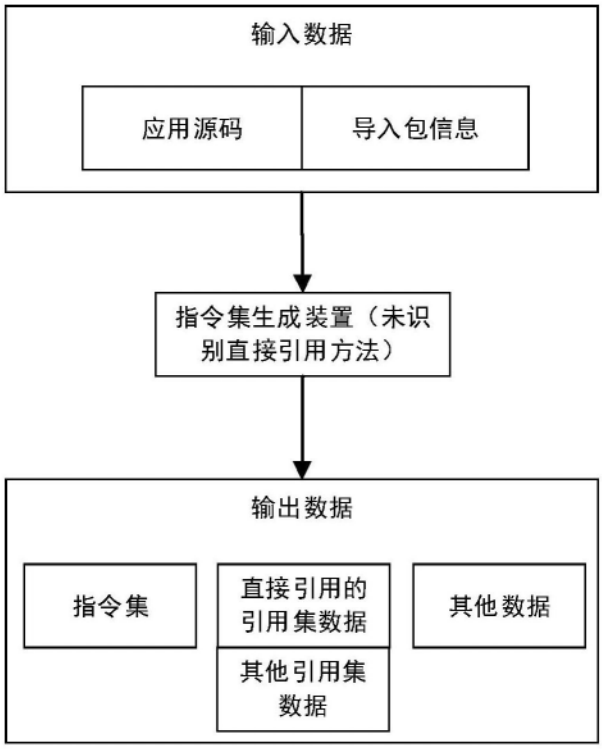

输入数据
应用源码
导入包信息
指令集生成装置（未识别直接引用方法）
输出数据
指令集
直接引用的引用集数据
其他引用集数据
其他数据

图2

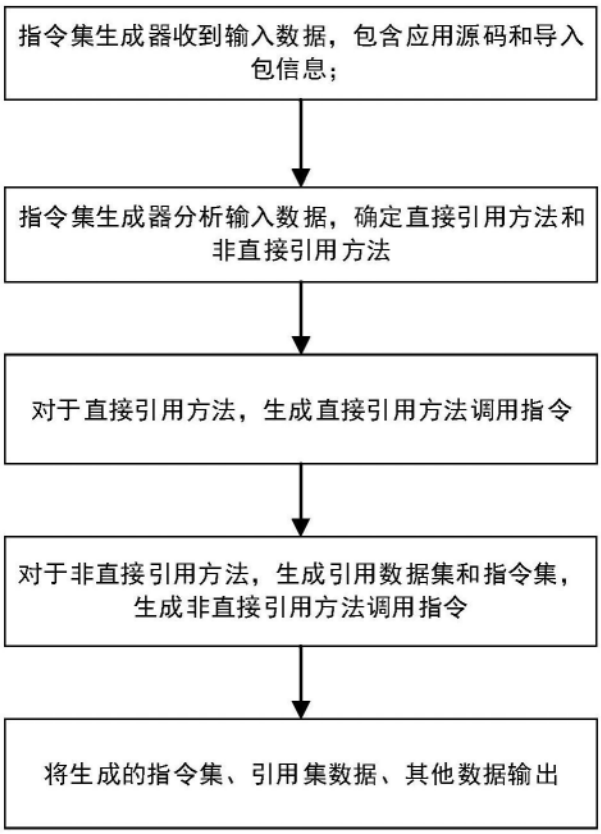

指令集生成器收到输入数据，包含应用源码和导入包信息；
指令集生成器分析输入数据，确定直接引用方法和非直接引用方法
对于直接引用方法，生成直接引用方法调用指令
对于非直接引用方法，生成引用数据集和指令集，生成非直接引用方法调用指令
将生成的指令集、引用集数据、其他数据输出

图3

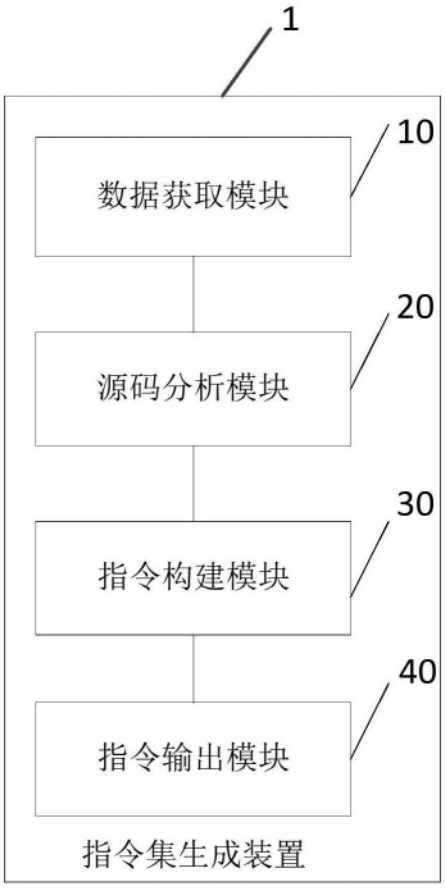

1
10
数据获取模块
20
源码分析模块
30
指令构建模块
40
指令输出模块
指令集生成装置

图4

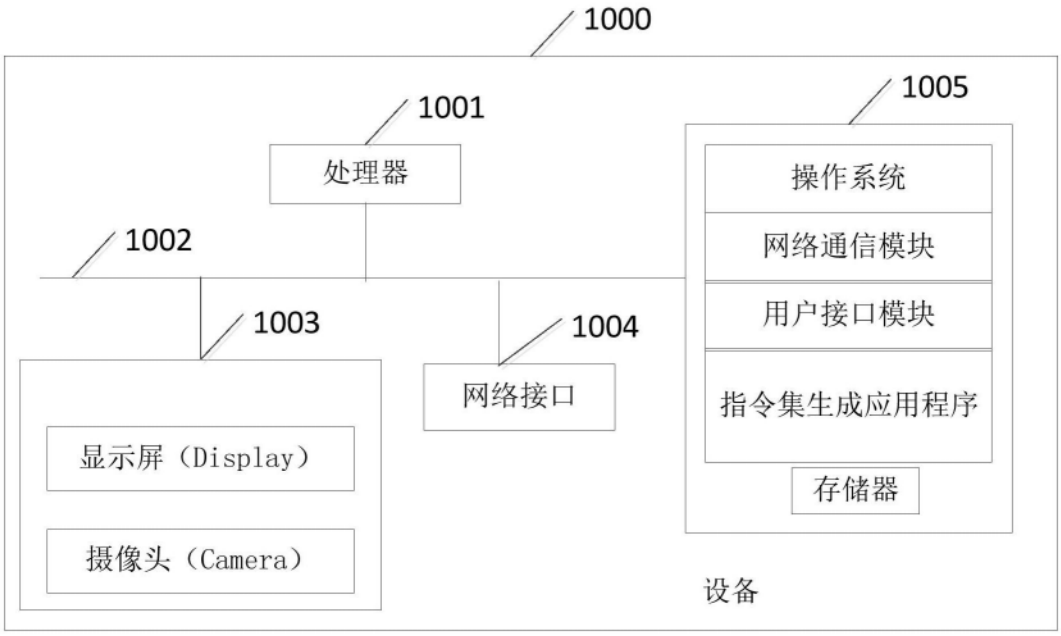

1000
1001
处理器
1002
1003
显示屏（Display）
摄像头（Camera）
1004
网络接口
1005
操作系统
网络通信模块
用户接口模块
指令集生成应用程序
存储器
设备

图5