(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 117195241 A
(43)申请公布日 2023.12.08

(21)申请号 202311476626.5

(22)申请日 2023.11.08

(71)申请人 蔚来汽车科技(安徽)有限公司
地址 230601 安徽省合肥市经济技术开发区宿松路3963号恒创智能科技园F幢
申请人 梅润元

(72)发明人 梅润元 王衍豪 程越强

(74)专利代理机构 北京瀚仁知识产权代理事务所(普通合伙) 11482
专利代理师 王国赛

(51)Int.Cl.
G06F 21/57(2013.01)
G06F 21/56(2013.01)
G06F 8/41(2018.01)

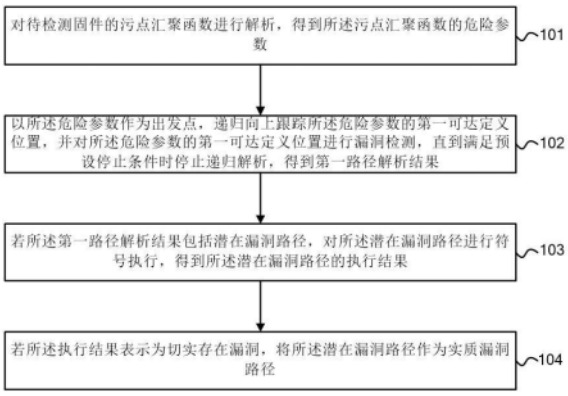
权利要求书3页 说明书14页 附图3页

(54)发明名称

固件漏洞的检测方法、设备和介质

(57)摘要

本申请提供了一种固件漏洞的检测方法、设备和介质,包括:对待检测固件的污点汇聚函数进行解析,得到污点汇聚函数的危险参数;以危险参数作为出发点,递归向上跟踪危险参数的第一可达定义位置,并进行漏洞解析,直到满足预设停止条件时停止递归解析,得到第一路径解析结果;若第一路径解析结果包括潜在漏洞路径,对潜在漏洞路径进行符号执行,若得到潜在漏洞路径的执行结果表示切实存在漏洞,将潜在漏洞路径作为实质漏洞路径。这样,由数据流进行潜在漏洞路径分析,利用符号执行对潜在漏洞路径验证控制流可达性,减少了符号执行的输入量,降低了路径爆炸风险,且数据流的潜在漏洞路径的分析过程相对简单,从而保证漏洞路径的检测精度和效率。



CN 117195241 A

1.一种固件漏洞的检测方法,其特征在于,包括:

对待检测固件的污点汇聚函数进行解析,得到所述污点汇聚函数的危险参数;

以所述危险参数作为出发点,递归向上跟踪所述危险参数的第一可达定义位置,并对所述危险参数的第一可达定义位置进行漏洞解析,直到满足预设停止条件时停止递归解析,得到第一路径解析结果;

若所述第一路径解析结果包括潜在漏洞路径,对所述潜在漏洞路径进行符号执行,得到所述潜在漏洞路径的执行结果;

若所述执行结果表示为切实存在漏洞,将所述潜在漏洞路径作为实质漏洞路径。

2.根据权利要求1所述的固件漏洞的检测方法,其特征在于,对所述危险参数的第一可达定义位置进行漏洞解析,直到满足预设停止条件时停止递归解析,得到第一路径解析结果,包括:

对于任意一次跟踪到的第一可达定义位置而言,确定所述第一可达定义位置对应的参数定义类型;

若所述参数定义类型能够触发所述待检测固件的污点源数据,确定满足预设停止条件,并停止递归解析,得到的所述第一路径解析结果为将跟踪过程中的递归路径作为所述潜在漏洞路径;

若所述参数定义类型被判定为不能到达所述污点源数据,确定满足预设停止条件,并停止递归解析,得到的所述第一路径解析结果为舍弃所述递归路径;

若所述参数定义类型既不能够触发所述污点源数据,又不能被判定为不能到达所述污点源数据,确定不满足预设停止条件,并继续进行递归解析。

3.根据权利要求2所述的固件漏洞的检测方法,其特征在于,还包括:

若所述参数定义类型为污点源定义类型,确定所述参数定义类型能够触发污点源数据;

若所述参数定义类型为常量定义类型、未初始化局部变量定义类型、涉及类型转换的函数返回值定义类型或者作用域范围越界的函数输入参数定义类型,将所述参数定义类型被判定为不能到达所述污点源数据;

若所述参数定义类型为局部变量定义类型、不涉及类型转换的函数返回值定义类型、格式化参数定义类型或作用域范围未越界的函数输入参数定义类型,确定所述参数定义类型既不能够触发所述污点源数据,又不能被判定为不能到达所述污点源数据。

4.根据权利要求1所述的固件漏洞的检测方法,其特征在于,以所述危险参数作为出发点,递归向上跟踪所述危险参数的第一可达定义位置,包括:

将所述危险参数转换成用于传递所述危险参数的载体信息;

在每次递归跟踪时,按照预设查找方式查找所述载体信息的位置,直到满足所述载体信息的位置的输出条件时,得到每次递归跟踪时得到的所述危险参数的第一可达定义位置;

所述预设查找方式包括:

检测在当次跟踪的基本块的查找位置是否为所跟踪函数的起始位置;

若所述查找位置为所述所跟踪函数的起始位置,且在所述当次跟踪的基本块内直到所述起始位置仍未查找到所述载体信息,输出所述所跟踪函数的起始位置为所述危险参数的

第一可达定义位置；

若所述查找位置不为所述所跟踪函数的起始位置，且在所述当次跟踪的基本块内未查找到所述载体信息，在所述所跟踪的函数内从所述当次跟踪的基本块的前驱基本块继续查找所述载体信息；

若所述查找位置不为所述所跟踪函数的起始位置，且在所述当次跟踪的基本块内查找到所述载体信息，检测所述危险参数的定义是否为条件赋值；

若所述危险参数的定义不为条件赋值，输出所述载体信息的位置作为所述危险参数的第一可达定义位置；

若所述危险参数的定义为条件赋值，在所述所跟踪的函数内从所述当次跟踪的基本块的前驱基本块继续查找所述载体信息。

5.根据权利要求1所述的固件漏洞的检测方法，其特征在于，对待检测固件的污点汇聚函数进行解析，得到所述污点汇聚函数的危险参数，包括：

在利用深度优先算法查找函数调用路径过程中，若检测到所述函数调用路径中当前的调用函数为污点汇聚函数，对所述待检测固件的污点汇聚函数进行解析，得到所述污点汇聚函数的危险参数；

其中，利用深度优先算法查找函数调用路径过程，包括：

对所述待检测固件进行预处理，得到所述待检测固件的污点源数据和目标程序；

获取所述污点源数据的引用函数；

基于所述待检测固件的目标程序和深度优先算法，以所述引用函数为出发点，递归向下跟踪所有子函数，得到所述引用函数对应的函数调用路径；其中，所述引用函数和所有子函数均作为所述函数调用路径中的调用函数。

6.根据权利要求5所述的固件漏洞的检测方法，其特征在于，还包括：

在利用深度优先算法查找函数调用路径过程中，当检测到所述当前的调用函数为库函数时，获取所述库函数的函数摘要；

若所述函数摘要表示存在危险，将所述库函数作为所述污点汇聚函数，并在污点汇聚函数列表中加载所述库函数；

若所述函数摘要表示不存在危险，将所述库函数不作为所述污点汇聚函数。

7.根据权利要求6所述的固件漏洞的检测方法，其特征在于，获取所述库函数的函数摘要，包括：

若所述库函数已经完成分析，获取所述库函数的既有函数摘要作为库函数的函数摘要；其中，所述既有函数摘要包括不存在危险或存在危险；

若所述库函数未完成分析，确定所述库函数所属的动态链接库的安全级别；

若所述动态链接库的安全级别高于或等于预设安全级别，获取所述库函数的函数摘要为不存在危险；

若所述动态链接库的安全级别低于预设安全级别，基于所述库函数与所述污点汇聚函数之间形成的库函数调用路径，以所述危险参数作为出发点，按照所述库函数调用路径递归向上跟踪所述危险参数的第二可达定义位置，直到满足预设停止条件时停止递归解析，得到第二路径解析结果，若所述第二路径解析结果包括存在潜在漏洞路径，获取所述库函数的函数摘要为存在危险，若所述第二路径解析结果包括不存在潜在漏洞路径，获取所述

库函数的函数摘要为不存在危险。

8.根据权利要求5所述的固件漏洞的检测方法,其特征在于,还包括:

在利用深度优先算法查找函数调用路径过程中,当所述当前的调度函数已被标记为局部安全函数时,停止对所述当前的调度函数的子函数进行追踪;其中,所述局部安全函数包括第一类安全函数和/或第二类安全函数;所述第一类安全函数为不能触及所述污点汇聚函数的函数,所述第二类安全函数为输入参数不会影响到所述污点汇聚函数的函数;

当所述当前的调度函数未被标记为局部安全函数,且所述当前的调度函数不是所述污点汇聚函数时,继续对所述当前的调度函数的子函数进行追踪,直到所述当前的调度函数的最后一个子函数仍不是所述污点汇聚函数时,将所述当前的调度函数标记为所述第一类安全函数,并进行剪枝;

当所述当前的调度函数未被标记为局部安全函数,且所述当前的调用函数不是所述污点汇聚函数时,继续对所述当前的调度函数的子函数进行追踪,直到所述当前的调度函数的最后一个子函数为所述污点汇聚函数时,若所述当前的调度函数的输入参数不影响所述污点汇聚函数,将所述当前的调度函数标记为所述第二类安全函数,并进行剪枝。

9.根据权利要求1所述的固件漏洞的检测方法,其特征在于,对所述潜在漏洞路径进行符号执行,得到所述潜在漏洞路径的执行结果,包括:

提取所述潜在漏洞路径的路径起点和路径终点,其中,所述路径起点为输入参数所对应的位置,所述路径终点为所述危险参数所对应的位置;

对所述输入参数进行漏洞注入,对所述潜在漏洞路径进行符号执行后,获取所述危险参数的返回数据;

若所述返回数据表示危险参数被污染,得到的所述执行结果为切实存在漏洞;

若所述返回数据表示危险参数未被污染,得到的所述执行结果为不存在漏洞。

10.一种固件漏洞的检测设备,其特征在于,包括处理器和存储装置,所述存储装置适于存储多条程序代码,所述程序代码适于由所述处理器加载并运行以执行权利要求1至9中任一项所述的固件漏洞的检测方法。

11.一种计算机可读存储介质,其特征在于,存储有多条程序代码,其特征在于,所述程序代码适于由处理器加载并运行以执行权利要求1至9中任一项所述的固件漏洞的检测方法。

固件漏洞的检测方法、设备和介质

技术领域

[0001] 本申请涉及物联网设备安全技术领域,具体提供一种固件漏洞的检测方法、设备和介质。

背景技术

[0002] 近年来,物联网行业快速发展,物联网设备的普及率越来越高。与此同时,物联网设备安全问题也变得愈发重要,路由器,智能音箱,智能门锁等物联网设备由于涉及用户的关键隐私信息,一旦遭到攻击者的入侵与操控,将会造成严重的安全隐患。针对物联网设备的攻击可以被划分为云、管、端三个攻击维度,云安全主要关注物联网厂商搭建的云平台的安全性,管安全主要关注通信协议的安全性,端安全主要关注终端设备的安全性。

[0003] 对于终点设备的安全性而言,早期的物联网设备以及其使用的应用协议由于安全性设计考虑不充分,存在各种脆弱性问题。此外,终端设备暴露在互联网上的数量庞大,又时常存在更新滞后的问题,大批量已经不再更新维护的终端,如果不经过有效治理,将长期存在脆弱性和风险。针对这一现状,国内外安全研究者在二进制固件漏洞挖掘领域进行了相应的研究。然而,与传统平台的漏洞发现相比,物联网终端设备的二进制固件漏洞发现存在着固件难以获取、固件格式差异较大、符号信息缺失、仿真困难、运行环境受限等挑战,如何对物联网设备固件进行高效且自动化的漏洞挖掘也受到了业界的广泛关注。

[0004] 通常情况下,二进制固件的漏洞发现方法可以被分为面向执行过程的动态分析方法以及面向二进制程序的静态分析方法两大类别。

[0005] 动态分析方法通过在计算机上仿真ARM与MIPS等嵌入式架构,并通过运行二进制固件中包含的固件文件系统模拟终端设备的执行,进而进行漏洞分析。然而,由于物联网设备多源异构,外设依赖较为复杂,各种仿真软件在仿真终端设备的外围设备以及处理启动配置信息等方面存在诸多缺陷,直接切换文件系统后,固件文件系统运行状态有时将与预期不符。因此,在仿真运行的基础上,FIRM-AFL等框架也通过模糊测试的方式,从动态分析的角度对二进制固件进行了自动化漏洞发现尝试。然而,一方面固件仿真运行存在诸多障碍,另一方面模糊测试方案也存在误报率高等问题。

[0006] 因此,以污点分析和符号执行为代表的二进制固件静态分析方案由于既可以在不具体运行程序的情况下发现漏洞,又从源代码层面具体解析了漏洞触发路径,解决了模糊测试误报率高的问题,受到了国内外研究者们的广泛关注。

[0007] 在现有研究工作中,可以使用符号执行结合数据流分析方法进行污点传播分析。然而,这种污点传播分析中,数据流分析方法通常用作剪枝,即剔除不需要处理的指令集合,然后利用符号执行同时实现数据流和控制流两个方面的跟踪,得到漏洞路径。也就是说,这种污点传播分析中数据流分析并不是发现漏洞路径的主导跟踪,仍然是以符号执行作为主导发现漏洞路径。

[0008] 然而,使用符号执行进行污点传播时通常需要针对不同类型的指令设置污点传播规则以及符号执行约束条件,从而在发现执行路径之后通过求解约束判断路径的可达性。

这一设计过程较为复杂,且由于物联网设备多源异构,设计复杂性进一步提高,这可能导致更多设计缺陷的产生,从而影响漏洞挖掘的精确性。

[0009] 因此,如何实现精确且高效的检测出漏洞路径是本领域技术人员亟待解决的技术问题。

发明内容

[0010] 为了克服上述缺陷,提出了本申请,以提供解决或至少部分地解决出漏洞路径检测效率、精度较低的技术问题的固件漏洞的检测方法、设备和介质。

[0011] 在第一方面,本申请提供一种固件漏洞的检测方法,该固件漏洞的检测方包括:

对待检测固件的污点汇聚函数进行解析,得到所述污点汇聚函数的危险参数;

以所述危险参数作为出发点,递归向上跟踪所述危险参数的第一可达定义位置,并对所述危险参数的第一可达定义位置进行漏洞解析,直到满足预设停止条件时停止递归解析,得到第一路径解析结果;

若所述第一路径解析结果包括潜在漏洞路径,对所述潜在漏洞路径进行符号执行,得到所述潜在漏洞路径的执行结果;

若所述执行结果表示为切实存在漏洞,将所述潜在漏洞路径作为实质漏洞路径。

[0012] 进一步地,上述所述的固件漏洞的检测方法中,对所述危险参数的第一可达定义位置进行漏洞解析,直到满足预设停止条件时停止递归解析,得到第一路径解析结果,包括:

对于任意一次跟踪到的第一可达定义位置而言,确定所述第一可达定义位置对应的参数定义类型;

若所述参数定义类型能够触发所述待检测固件的污点源数据,确定满足预设停止条件,并停止递归解析,得到的所述第一路径解析结果为将跟踪过程中的递归路径作为所述潜在漏洞路径;

若所述参数定义类型被判定为不能到达所述污点源数据,确定满足预设停止条件,并停止递归解析,得到的所述第一路径解析结果为舍弃所述递归路径;

若所述参数定义类型既不能够触发所述污点源数据,又不能被判定为不能到达所述污点源数据,确定不满足预设停止条件,并继续进行递归解析。

[0013] 进一步地,上述所述的固件漏洞的检测方法,还包括:

若所述参数定义类型为污点源定义类型,确定所述参数定义类型能够触发污点源数据;

若所述参数定义类型为常量定义类型、未初始化局部变量定义类型、涉及类型转换的函数返回值定义类型或者作用域范围越界的函数输入参数定义类型,将所述参数定义类型被判定为不能到达所述污点源数据;

若所述参数定义类型为局部变量定义类型、不涉及类型转换的函数返回值定义类型、格式化参数定义类型或作用域范围未越界的函数输入参数定义类型,确定所述参数定义类型既不能够触发所述污点源数据,又不能被判定为不能到达所述污点源数据。

[0014] 进一步地,上述所述的固件漏洞的检测方法中,以所述危险参数作为出发点,递归向上跟踪所述危险参数的第一可达定义位置,包括:

将所述危险参数转换成用于传递所述危险参数的载体信息；

在每次递归跟踪时，按照预设查找方式查找所述载体信息的位置，直到满足所述载体信息的位置的输出条件时，得到每次递归跟踪时得到的所述危险参数的第一可达定义位置；

所述预设查找方式包括：

检测在当次跟踪的基本块的查找位置是否为所跟踪函数的起始位置；

若所述查找位置为所述所跟踪函数的起始位置，且在所述当次跟踪的基本块内直到所述起始位置仍未查找到所述载体信息，输出所述所跟踪函数的起始位置为所述危险参数的第一可达定义位置；

若所述查找位置不为所述所跟踪函数的起始位置，且在所述当次跟踪的基本块内未查找到所述载体信息，在所述所跟踪的函数内从所述当次跟踪的基本块的前驱基本块继续查找所述载体信息；

若所述查找位置不为所述所跟踪函数的起始位置，且在所述当次跟踪的基本块内查找到所述载体信息，检测所述危险参数的定义是否为条件赋值；

若所述危险参数的定义不为条件赋值，输出所述载体信息的位置作为所述危险参数的第一可达定义位置；

若所述危险参数的定义为条件赋值，在所述所跟踪的函数内从所述当次跟踪的基本块的前驱基本块继续查找所述载体信息。

[0015] 进一步地，上述所述的固件漏洞的检测方法中，对待检测固件的污点汇聚函数进行解析，得到所述污点汇聚函数的危险参数，包括：

在利用深度优先算法查找函数调用路径过程中，若检测到所述函数调用路径中当前的调用函数为污点汇聚函数，对待检测固件的污点汇聚函数进行解析，得到所述污点汇聚函数的危险参数；

其中，利用深度优先算法查找函数调用路径过程，包括：

对所述待检测固件进行预处理，得到所述待检测固件的污点源数据和目标程序；

获取所述污点源数据的引用函数；

基于所述待检测固件的目标程序和深度优先算法，以所述引用函数为出发点，递归向下跟踪所有子函数，得到所述引用函数对应的函数调用路径；其中，所述引用函数和所有子函数均作为所述函数调用路径中的调用函数。

[0016] 进一步地，上述所述的固件漏洞的检测方法，还包括：

在利用深度优先算法查找函数调用路径过程中，当检测到所述当前的调用函数为库函数时，获取所述库函数的函数摘要；

若所述函数摘要表示存在危险，将所述库函数作为所述污点汇聚函数，并在污点汇聚函数列表中加载所述库函数；

若所述函数摘要表示不存在危险，将所述库函数不作为所述污点汇聚函数。

[0017] 进一步地，上述所述的固件漏洞的检测方法中，获取所述库函数的函数摘要，包括：

若所述库函数已经完成分析，获取所述库函数的既有函数摘要作为库函数的函数摘要；其中，所述既有函数摘要包括不存在危险或存在危险；

若所述库函数未完成分析，确定所述库函数所属的动态链接库的安全级别；

若所述动态链接库的安全级别高于或等于预设安全级别，获取所述库函数的函数摘要为不存在危险；

若所述动态链接库的安全级别低于预设安全级别，基于所述库函数与所述污点汇聚函数之间形成的库函数调用路径，以所述危险参数作为出发点，按照所述库函数调用路径递归向上跟踪所述危险参数的第二可达定义位置，直到满足预设停止条件时停止递归解析，得到第二路径解析结果，若所述第二路径解析结果包括存在潜在漏洞路径，获取所述库函数的函数摘要为存在危险，若所述第二路径解析结果包括不存在潜在漏洞路径，获取所述库函数的函数摘要为不存在危险。

[0018] 进一步地，上述所述的固件漏洞的检测方法，还包括：

在利用深度优先算法查找函数调用路径过程中，当所述当前的调度函数已被标记为局部安全函数时，停止对所述当前的调度函数的子函数进行追踪；其中，所述局部安全函数包括第一类安全函数和/或第二类安全函数；所述第一类安全函数为不能触及所述污点汇聚函数的函数，所述第二类安全函数为输入参数不会影响到所述污点汇聚函数的函数；

当所述当前的调度函数未被标记为局部安全函数，且所述当前的调度函数不是所述污点汇聚函数时，继续对所述当前的调度函数的子函数进行追踪，直到所述当前的调度函数的最后一个子函数仍不是所述污点汇聚函数时，将所述当前的调度函数标记为所述第一类安全函数，并进行剪枝；

当所述当前的调度函数未被标记为局部安全函数，且所述当前的调用函数不是所述污点汇聚函数时，继续对所述当前的调度函数的子函数进行追踪，直到所述当前的调度函数的最后一个子函数为所述污点汇聚函数时，若所述当前的调度函数的输入参数不影响所述污点汇聚函数，将所述当前的调度函数标记为所述第二类安全函数，并进行剪枝。

[0019] 进一步地，上述所述的固件漏洞的检测方法中，对所述潜在漏洞路径进行符号执行，得到所述潜在漏洞路径的执行结果，包括：

提取所述潜在漏洞路径的路径起点和路径终点，其中，所述路径起点为输入参数所对应的位置，所述路径终点为所述危险参数所对应的位置；

对所述输入参数进行漏洞注入，对所述潜在漏洞路径进行符号执行后，获取所述危险参数的返回数据；

若所述返回数据表示危险参数被污染，得到的所述执行结果为切实存在漏洞；

若所述返回数据表示危险参数未被污染，得到的所述执行结果为不存在漏洞。

[0020] 在第二方面，本申请提供一种固件漏洞的检测设备，包括处理器和存储装置，所述存储装置适于存储多条程序代码，所述程序代码适于由所述处理器加载并运行以执行上述任一项所述的固件漏洞的检测方法。

[0021] 在第三方面，提供一种计算机可读存储介质，其特征在于，存储有多条程序代码，其特征在于，所述程序代码适于由处理器加载并运行以执行上述任一项所述的固件漏洞的检测方法。

[0022] 本申请上述一个或多个技术方案，至少具有如下一种或多种有益效果：

在实施本申请的技术方案中，通过对待检测固件的污点汇聚函数进行解析，得到所述污点汇聚函数的危险参数后，以所述危险参数作为出发点，递归向上跟踪所述危险参

数的第一可达定义位置,并对所述危险参数的第一可达定义位置进行漏洞解析,直到满足预设停止条件时停止递归解析,得到第一路径解析结果,若所述第一路径解析结果包括潜在漏洞路径,对所述潜在漏洞路径进行符号执行,得到所述潜在漏洞路径的执行结果;若所述执行结果表示为切实存在漏洞,将所述潜在漏洞路径作为实质漏洞路径。这样,可以由数据流进行潜在漏洞路径分析,并利用符号执行对潜在漏洞路径进行复核,验证控制流可达性,减少了符号执行的输入量,降低了符号执行的路径爆炸风险,且数据流的潜在漏洞路径的分析过程相对简单,从而可以保证漏洞路径的检测精度和效率。

附图说明

[0023] 参照附图,本申请的公开内容将变得更易理解。本领域技术人员容易理解的是:这些附图仅仅用于说明的目的,而并非意在对本申请的保护范围组成限制。此外,图中类似的数字用以表示类似的部件,其中:

图1是根据本申请的一个实施例的固件漏洞的检测方法的主要步骤流程示意图;

图2是参数定义位置解析原理图;

图3是定义可达性反向污点跟踪的原理图;

图4是基于符号执行的控制流复核原理图;

图5是污点汇聚函数的扩展原理图;

图6是根据本申请的一个实施例的固件漏洞的检测设备的主要结构框图。

具体实施方式

[0024] 下面参照附图来描述本申请的一些实施方式。本领域技术人员应当理解的是,这些实施方式仅仅用于解释本申请的技术原理,并非旨在限制本申请的保护范围。

[0025] 在本申请的描述中,“模块”、“处理器”可以包括硬件、软件或者两者的组合。一个模块可以包括硬件电路,各种合适的感应器,通信端口,存储器,也可以包括软件部分,比如程序代码,也可以是软件和硬件的组合。处理器可以是中央处理器、微处理器、图像处理器、数字信号处理器或者其他任何合适的处理器。处理器具有数据和/或信号处理功能。处理器可以以软件方式实现、硬件方式实现或者二者结合方式实现。非暂时性的计算机可读存储介质包括任何合适的可存储程序代码的介质,比如磁碟、硬盘、光碟、闪存、只读存储器、随机存取存储器等等。术语“A和/或B”表示所有可能的A与B的组合,比如只是A、只是B或者A和B。术语“至少一个A或B”或者“A和B中的至少一个”含义与“A和/或B”类似,可以包括只是A、只是B或者A和B。单数形式的术语“一个”、“这个”也可以包含复数形式。

[0026] 参阅附图1,图1是根据本申请的一个实施例的固件漏洞的检测方法的主要步骤流程示意图。如图1所示,本申请实施例中的固件漏洞的检测方法主要包括下列步骤101-步骤104。

[0027] 步骤101、对待检测固件的污点汇聚函数进行解析,得到所述污点汇聚函数的危险参数;

在一个具体实现过程中,可以对待检测固件进行预处理获得待检测固件的目标程序和污点源数据。然后基于预先规定的需要关注的污点汇聚函数列表,对目标程序中的函数进行解析,得到待检测固件的污点汇聚函数,并进一步对待检测固件的污点汇聚函数进

行解析得到所述污点汇聚函数的危险参数。其中,待检测固件的污点汇聚函数为直接产生安全敏感操作或者泄露隐私数据到外界的函数。污点源数据的查找可以以提示用户输入信息位置的关键字字符串为媒介进行查找,这里可以通过交叉引用查找定位位于数据节的关键字的程序引用位置,再在这些污点源引用位置处进行分析得到污点源数据,其中,污点源数据为直接引入不受信任的数据或者机密数据到系统中的数据,如用户输入的网址等。

[0028] 步骤102、以所述危险参数作为出发点,递归向上跟踪所述危险参数的第一可达定义位置,并对所述危险参数的第一可达定义位置进行漏洞检测,直到满足预设停止条件时停止递归解析,得到第一路径解析结果;

在一个具体实现过程中,在得到污点汇聚函数的危险参数后,可以从参数使用位置到参数定义位置进行递归向上跟踪,检测程序中危险参数的参数定义位置作为第一可达定义位置,并对所述危险参数的第一可达定义位置进行是否存在被敏感信息控制的可能,实现在危险参数的第一可达定义位置进行漏洞解析,以发现潜在漏洞路径,直到满足预设停止条件时停止递归解析,得到第一路径解析结果,这样,可以在数据流上得到哪些危险参数的参数定义位置可以达到程序中参数使用位置,并从危险参数的参数定义位置分析其是否可能被敏感信息控制实现潜在漏洞路径的发现。从而得到数据流中哪些路径可能是潜在漏洞路径,哪些路径不是潜在漏洞路径。

[0029] 具体地,图2是参数定义位置解析原理图。如图2所示,在进行参数定义位置解析时,可以将所述危险参数转换成用于传递所述危险参数的载体信息(如寄存器的编码或内存使用情况等),并基于固件的目标程序生成的控制流图,在每次递归跟踪时,按照预设查找方式查找所述载体信息的位置,直到满足所述载体信息的位置的输出条件时,得到每次递归跟踪时得到的所述危险参数的第一可达定义位置。

[0030] 其中,预设查找方式包括如下操作:

(1)在当次跟踪的基本块内查找载体信息时,检测在当次跟踪的基本块的查找位置是否为所跟踪的函数的起始位置;

在一个具体实现过程中,对于跟踪的函数而言,如果到达所跟踪的函数的起始位置,仍未查找到危险参数的载体信息,则说明危险参数的定义来源于上一层函数或者未初始化局部变量定义,此时,可以在当前所跟踪的函数中停止查找,因此,需要检测在当次跟踪的基本块的查找位置是否为所跟踪的函数的起始位置。

[0031] (2)若所述查找位置为所述所跟踪函数的起始位置,且在所述当次跟踪的基本块内直到所述起始位置仍未查找到所述载体信息,输出所述所跟踪函数的起始位置为所述危险参数的第一可达定义位置;

在一个具体实现过程中,若所述查找位置为所述所跟踪函数的起始位置,但所述查找位置未查找到所述载体信息,说明危险参数的定义来源于上一层函数或者未初始化局部变量定义,此时,可以在当前所跟踪的函数中停止查找,直接将所跟踪函数的起始位置为所述危险参数的第一可达定义位置作为危险参数的第一可达定义位置以便后续对危险参数的第一可达定义位置进行漏洞分析。

[0032] (3)若所述查找位置不为所述所跟踪函数的起始位置,且所述当次跟踪的基本块内未查找到所述载体信息,在所述所跟踪的函数内从所述当次跟踪的基本块的前驱基本块继续查找所述载体信息;

在一个具体实现过程中,若所述查找位置不为所述所跟踪函数的起始位置,且所述当次跟踪的基本块内未查找到所述载体信息,在所述所跟踪的函数内从所述当次跟踪的基本块的前驱基本块继续查找所述载体信息。这样,对于循环而言,参数定义位置位于参数使用位置后方时,危险参数的定义的赋值依赖于循环体中最后一个基本块的前驱节点指向循环起点,而循环体中最后一个基本块的前驱节点指向循环起点,因此寻找前驱节点过程中程序能够正确解析循环情况,判断位于参数使用位置后方的参数定义位置可以影响到参数使用位置。

[0033] (4)若所述查找位置不为所述所跟踪函数的起始位置,且所述当次跟踪的基本块内查找到所述载体信息,检测所述危险参数的定义是否为条件赋值;

在一个具体实现过程中,危险参数的定义可能会受到赋值情况的影响,例如,对于分支情况而言,如果某一危险参数的定义的赋值位于分支之中,即该基本块的前驱基本块不唯一,则所有前驱基本块内的定义语句都有可能影响该处危险参数的使用位置,即所有分支中危险参数的定义情况都需要进行输出,这种情况下,危险参数的定义的赋值为条件赋值(非确定定义赋值)。因此,若所述查找位置不为所述所跟踪函数的起始位置,且所述当次跟踪的基本块内查找到所述载体信息,可以检测所述危险参数的定义是否为条件赋值。

[0034] (5)若所述危险参数的定义不为条件赋值,输出所述载体信息的位置作为所述危险参数的第一可达定义位置;

(6)若所述危险参数的定义为条件赋值,在所述所跟踪的函数内从所述当次跟踪的基本块的前驱基本块继续查找所述载体信息。

[0035] 在一个具体实现过程中,图3是定义可达性反向污点跟踪的原理图,如图3所示,污点汇聚函数输入后,对污点汇聚函数进行解析能够得到污点汇聚函数的危险参数,作为解析起点,然后递归向上追踪,解析危险参数定义位置,得到每次追踪到的第一可达定义位置并进行漏洞解析得到第一可达定义位置对应的参数定义类型作为解析结果,以便根据解析结果,判断是否满足预设停止条件,直到判断出满足预设停止条件时停止递归解析,得到第一路径解析结果。具体的操作步骤可以包括如下步骤:

(11)对于任意一次跟踪到的第一可达定义位置而言,确定所述第一可达定义位置对应的参数定义类型;

在一个具体实现过程中,由于定义可达性分析是一种跟踪单一变量定义位置的分析方式,在用于污点传播时需要从作为污点汇聚点函数的危险参数出发,递归向上跟踪危险参数的定义位置,直到危险参数的定义来源触发污点源数据,或被判定为不可能到达污点源数据为止。其中,当以下两种情况发生时,这里可以判断危险参数无法到达污点源数据,从而停止递归过程:

a、常量定义类型、涉及类型转换的函数返回值定义类型或未初始化局部变量定义类型:当危险参数的内容来源于常量或未初始化局部变量,其内容来源将无法被其他变量影响。当危险参数定义来源于涉及类型转换的函数返回值时,由于数据类型受限,其将无法导致缓冲区溢出或命令注入漏洞。

[0036] b、作用域范围越界:在污点源数据的变量作用域范围内无法定位到任何污点源数据对于危险参数的影响语句。由于单一变量只能在其作用域范围内使用,一旦作用域范围越界,则污点源数据变量不可能再对危险参数产生影响,此时应当停止继续分析。

[0037] 在一个具体实现过程中,在参数定义位置的解析时,定义可达性分析方案可能遇到多种类型的参数定义情况。对于不同定义情况的数据流解析是本方案进行漏洞解析的主要依据,因此,可以对任意一次跟踪到的第一可达定义位置,确定所述第一可达定义位置对应的参数定义类型,如可以根据第一可达定义位置对应的程序代码得到第一可达定义位置对应的参数定义类型。第一可达定义位置对应的参数定义类型可以包括常量定义类型、未初始化局部变量定义类型、涉及类型转换的函数返回值定义类型、作用域范围越界的函数输入参数定义类型、局部变量定义类型、不涉及类型转换的函数返回值定义类型、格式化参数定义类型、作用域范围未越界的函数输入参数定义类型和污点源定义类型等。

[0038] (12)若所述参数定义类型能够触发所述待检测固件的污点源数据,确定满足预设停止条件,并停止递归解析,得到的所述第一路径解析结果为将跟踪过程中的递归路径作为所述潜在漏洞路径;

在一个具体实现过程中,若所述参数定义类型为污点源定义类型,说明污点源数据与污点汇聚函数具有依赖关系,可以确定所述参数定义类型能够触发污点源数据,已满足预设停止条件,并停止递归解析,得到的所述第一路径解析结果为将跟踪过程中的递归路径作为所述潜在漏洞路径。

[0039] (13)若所述参数定义类型被判定为不能到达所述污点源数据,确定满足预设停止条件,并停止递归解析,得到的所述第一路径解析结果为舍弃所述递归路径;

在一个具体实现过程中,若所述参数定义类型为常量定义类型、未初始化局部变量定义类型、涉及类型转换的函数返回值定义类型或者作用域范围越界的函数输入参数定义类型,将所述参数定义类型被判定为不能到达所述污点源数据,已满足预设停止条件,并停止递归解析,得到的所述第一路径解析结果为舍弃所述递归路径。

[0040] 对于涉及类型转换的函数返回值定义类型而言,可以为图3中atoi等涉及类型转换函数。如果危险参数的定义来源于某一其他函数的返回值,则该危险参数的定义来源的值可能受到对应函数的所有参数的控制与影响。为了提高分析的覆盖率,本实施例在该情况下将函数所有参数都标记为了可以影响目标参数的上层定义来源,在此基础上继续递归解析这些参数的定义可达性,但是如果对应函数涉及字符串至整数、字符串至IP地址结构体等类型转换,则字符串无法继续导致溢出或命令注入漏洞,也就是说,该参数定义类型属于涉及类型转换的函数返回值定义类型,此时应当直接舍弃该次定义,进而舍弃所述递归路径。

[0041] 对于作用域范围越界的函数输入参数定义类型而言,函数输入参数定义对应的函数已经在函数调用路径上的起始函数之前,则该参数定义类型属于作用域范围越界的函数输入参数定义类型,此时,危险参数不会被其所在函数的输入参数影响,直接舍弃该次定义,不在进行递归解析。

[0042] (14)若所述参数定义类型既不能够触发所述污点源数据,又不能被判定为不能到达所述污点源数据,确定不满足预设停止条件,并继续进行递归解析。

[0043] 若所述参数定义类型为局部变量定义类型、不涉及类型转换的函数返回值定义类型、格式化参数定义类型或作用域范围未越界的函数输入参数定义类型,确定所述参数定义类型既不能够触发所述污点源数据,又不能被判定为不能到达所述污点源数据。

[0044] 对于局部变量定义类型而言,由于函数内部的局部变量的值一般情况下均来自于

常量或函数输入参数,因此,局部变量定义需要继续递归解析其定义来源到底是常量,还是函数输入参数,从而确定是否能够触发所述污点源数据。

[0045] 对于不涉及类型转换的函数返回值定义类型而言,如果危险参数的定义来源于某一其他函数的返回值,则该危险参数的定义来源的值可能受到对应函数的所有参数的控制与影响。为了提高分析的覆盖率,本实施例在该情况下将函数所有参数都标记为了可以影响目标参数的上层定义来源,如果对应函数不涉及字符串至整数、字符串至IP地址结构体等类型转换,也就是说,该参数定义类型属于不涉及类型转换的函数返回值定义类型,在此基础上继续递归解析这些参数的定义可达性。

[0046] 对于格式化参数定义类型而言,存在格式化字符串参数的危险函数,其输入参数个数往往是可变的。因此,需要通过查找其格式化字符串参数的定义情况确认该函数可能导致漏洞的危险参数定义位置。具体而言,若格式化字符串参数为变量定义,则判断其是否可能被污点源影响,若为常量定义,则为检测缓冲区溢出类型漏洞,需根据常量字符串中的“%s”字符串参数的个数与位置,将对应位置的其他输入参数纳入待检测的危险参数列表进行递归解析处理。

[0047] 对于作用域范围未越界的函数输入参数定义类型而言,如果函数输入参数定义对应的函数已经在函数调用路径上的起始函数后或者函数输入参数定义对应的函数为函数调用路径上的起始函数,则该参数定义类型属于作用域范围未越界的函数输入参数定义类型,此时,危险参数可能被其所在函数的输入参数影响,则需要根据函数调用路径作为启发式信息,以提示定位上层函数的调用位置,从而在上层函数中查找输入参数的定义可达性,在上层函数内部继续递归进行定义可达性解析。

[0048] 步骤103、若所述第一路径解析结果包括潜在漏洞路径,对所述潜在漏洞路径进行符号执行,得到所述潜在漏洞路径的执行结果;

步骤104、若所述执行结果表示为切实存在漏洞,将所述潜在漏洞路径作为实质漏洞路径。

[0049] 在一个具体实现过程中,得到的第一路径解析结果中包括潜在漏洞路径,但是该潜在漏洞路径仅能表明在数据流上可达,而实际中在控制流中并不可达,因此,若所述第一路径解析结果包括潜在漏洞路径,对所述潜在漏洞路径进行符号执行,得到所述潜在漏洞路径的执行结果,

具体地,图4是基于符号执行的控制流复核原理图。如图4所示,可以提取所述潜在漏洞路径的路径起点和路径终点,其中,所述路径起点为输入参数所对应的位置,所述路径终点为所述危险参数所对应的位置。对所述输入参数进行漏洞注入,生成初始状态,然后对所述潜在漏洞路径进行符号执行后,获取所述危险参数的返回数据,以便确定潜在漏洞路径是否在控制流可达,进行控制流不可达情况筛除;若所述返回数据表示危险参数被污染,得到的所述执行结果为切实存在漏洞,若所述返回数据表示危险参数未被污染,得到的所述执行结果为不存在漏,这样,可以在数据流与控制流两个方面筛选出完备的可达性的潜在漏洞函数,增强技术的自动化程度与完整度。

[0050] 例如,可以注入漏洞可以包括过长输入数据,完成符号执行后,危险参数的返回数据也为过长输入数据,则说明危险参数被污染,切实存在漏洞,否则,危险参数的返回数据恢复为较短的标准数据,则说明危险参数未被污染,不存在漏洞。

[0051] 再例如,可以注入含有控制命令的模式字符串,危险参数的返回数据未表现出报错或者过滤,则说明危险参数被污染,切实存在漏洞,否则,危险参数的返回数据表现为报错或过滤,则说明危险参数未被污染,不存在漏洞。

[0052] 在一个具体实现过程中,本实施例中由于直接对输入参数进行漏洞注入,即可进行符号执行,得到危险参数的返回数据,这样,无需对不同类型的指令设置污点传播规则以及符号执行约束条件,且无需进行约束求解,简化了符号执行的设计过程,提高了符号执行效率。且由于已经利用数据流进行了潜在漏洞的解析,在利用符号执行时,输入数据量相对较少,降低了符号执行的路径爆炸风险。

[0053] 本实施例的固件漏洞的检测方法,通过对待检测固件的污点汇聚函数进行解析,得到所述污点汇聚函数的危险参数后,以所述危险参数作为出发点,递归向上跟踪所述危险参数的第一可达定义位置,并对所述危险参数的第一可达定义位置进行漏洞解析,直到满足预设停止条件时停止递归解析,得到第一路径解析结果,若所述第一路径解析结果包括潜在漏洞路径,对所述潜在漏洞路径进行符号执行,得到所述潜在漏洞路径的执行结果;若所述执行结果表示为切实存在漏洞,将所述潜在漏洞路径作为实质漏洞路径。这样,可以由数据流进行潜在漏洞路径分析,并利用符号执行对潜在漏洞路径进行复核,验证控制流可达性,减少了符号执行的输入量,降低了符号执行的路径爆炸风险,且数据流的潜在漏洞路径的分析过程相对简单,从而可以保证漏洞路径的检测精度和效率。

[0054] 在一个具体实现过程中,由于定义可达性分析是一种从参数使用位置到参数定义位置的递归向上定义查找,如果不限制作用范围,该分析可能由于递归层数过深出现路径爆炸的问题。如果在不进行预处理的情况下将定义可达性分析扩展到程序范围,则当需要查找某个函数输入参数的定义位置时,因为无法判定位于何处的上层调用位置可能触及用户输入,程序必须遍历当前函数的所有调用位置,并在所有上层调用位置处分别进行进一步地定义可达性查找,这造成了很多不必要的路径查找。

[0055] 因此,本实施例中,对于每一个污点源数据而言,可以获取所述污点源数据的引用函数,本方法将通过查找函数间调用关系递归向下跟踪该引用函数所调用的所有子函数,并以深度优先算法的方式在这些子函数中再次进行函数调用路径分析,从而找到从污点源数据到污点汇聚点函数的函数调用路径,在定义可达性分析寻找函数上层调用位置时为其提供决定向何处上层调用位置分析的启发式信息,也就是图3中针对函数输入参数定义类型,在上层函数中查找调用函数对应输入参数的定义可达性时提供启发式信息。

[0056] 在一个具体实现过程中,利用深度优先算法查找函数调用路径过程,可以包括如下步骤:

(21)对所述待检测固件进行预处理,得到所述待检测固件的污点源数据和目标程序;

(22)获取所述污点源数据的引用函数;其中,可以通过交叉引用的方式获取污点源数据的引用函数。

[0057] (23)基于所述固件的目标程序和深度优先算法,以所述引用函数为出发点,递归向下跟踪所有子函数,得到所述引用函数对应的函数调用路径;其中,所述引用函数和所有子函数均作为所述函数调用路径中的调用函数。

[0058] 具体地,可以将待检测固件的目标程序转化成控制流图,然后基于控制流图和深

度优先算法,以所述引用函数为出发点,递归向下跟踪所有子函数,得到所述引用函数对应的函数调用路径。

[0059] 需要说明的是,也可以将引用函数作为终点,进行反向递归,得到所述引用函数对应的函数调用路径,但是递归分析的入口方面,作为正向递归入口的用户输入位置,相较于作为反向递归入口的危险函数(包括system,strcpy等常用函数)所有程序引用位置而言相对较少,更加适合作为递归起点。因此,这里选择使用正向递归方法进行函数调用路径分析。

[0060] 在一个具体实现过程中,在利用深度优先算法查找函数调用路径过程中,会对每次跟踪的调用函数进行识别得到每次跟踪的调用函数的类别,然后基于每次跟踪的调用函数的类别,进行分析,最终得到引用函数对应的函数调用路径。

[0061] 具体地,若检测到所述函数调用路径中当前的调用函数为污点汇聚函数,则可以对待检测固件的污点汇聚函数进行解析,得到所述污点汇聚函数的危险参数,以便进行定义可达性反向污点跟踪方法进行分析(详见步骤102至步骤104),在分析过程中已分析过的函数的漏洞可达状态信息,以便将这一记录信息被用于剪枝处理,避免对于单一函数的重复分析。

[0062] 在一个具体实现过程中,由于物联网设备架构多样,标准并不统一,除了使用常见第三方动态链接库外,物联网设备厂商也会自行制作厂商自定义的动态链接库程序,这些程序内部也可能存在潜在漏洞。例如,CVE-2021-41653 、CVE-2022-25060、CVE-2022-25061等漏洞就都位于TP-Link路由器设备的libcmm.so动态链接库中,该动态链接库由于含有未限制的命令执行调用,可能导致命令注入类型的漏洞。

[0063] 另一方面,由于二进制程序引入的动态链接库数量往往较多,对于网络边际程序所引入的所有动态链接库都进行加载与分析可能带来大量的运行时间开销,因此,前沿自动化漏洞挖掘分析框架往往只对strcpy等libc函数中常见的字符串处理函数生成函数摘要,并不对目标程序引入的其他动态链接库函数进行额外处理。这一处理方式与其使用的污点传播方法也有一定联系,基于符号执行的分析框架往往需要限制分析作用范围,避免产生路径爆炸。

[0064] 由于运用了定义可达性分析方法,本系统不使用时间开销过大的符号执行方法进行污点传播,因此有能力对库函数内部触发的漏洞进行检测。因此,本系统集成了一个污点汇聚点扩展模块,其为污点分析漏洞查找的过程中对于漏洞识别范围的一项扩展,通过将可能触发漏洞的自定义动态链接库函数纳入污点汇聚点的识别范畴,扩展本申请的漏洞识别范围。

[0065] 因此,在利用深度优先算法查找函数调用路径过程中,当检测到所述函数调用路径中当前的调用函数为库函数时,获取所述库函数的函数摘要;若所述函数摘要表示存在危险,将所述库函数作为所述污点汇聚函数,并在污点汇聚函数列表中加载所述库函数,以对污点汇聚函数列表进行扩展;若所述函数摘要表示不存在危险,将所述库函数不作为所述污点汇聚函数,不再对污点汇聚函数列表进行扩展。

[0066] 图5是污点汇聚函数的扩展原理图。如图5所述,当检测到所述函数调用路径中任一调用函数为库函数时,可以确定该库函数是否已经完成分析,若所述库函数已经完成分析,说明该库函数已经存在既有函数摘要,这样,可以直接获取所述库函数的既有函数摘要

作为库函数的函数摘要;其中,所述既有函数摘要包括不存在危险或存在危险;若所述库函数未完成分析,可以通过库函数的定义位置,确定所述库函数所属的动态链接库的安全级别(如图中是否属于第三方库);若所述动态链接库的安全级别高于或等于预设安全级别(如果该库函数并不位于libc,libcrypto等经过广泛测试,含有漏洞的可能性较低的第三方动态链接库中,而是位于厂商自行实现的动态链接库中,则可以确定库函数的安全级别低于预设安全级别,反之,则高于或等于预设安全级别),获取所述库函数的函数摘要为不存在危险;若所述动态链接库的安全级别低于预设安全级别,基于所述库函数与所述污点汇聚函数之间形成的库函数调用路径,以所述危险参数作为出发点,按照所述库函数调用路径递归向上跟踪所述危险参数的第二可达定义位置,直到满足预设停止条件时停止递归解析,得到第二路径解析结果,若所述第二路径解析结果包括存在潜在漏洞路径,获取所述库函数的函数摘要为存在危险,若所述第二路径解析结果包括不存在潜在漏洞路径,获取所述库函数的函数摘要为不存在危险。

[0067] 需要说明的是,对于该库函数而言,如果发现该库函数调用其他库函数,则基于污点汇聚函数的扩展原理自身进行调度即可,在此不再说明。

[0068] 本实施例中,能够通过库函数的动态加载机制进行加载,避免了分析全部库函数所带来的巨额时间开销,只有当系统判断污点源数据可能被某一库函数影响时,才会对该函数进行分析。当分析结束后,通过对于分析结果的记录,也可以避免对于单一函数的重复分析。

[0069] 在一个具体实现过程中,在利用深度优先算法查找函数调用路径过程中,当所述当前的调度函数已被标记为局部安全函数时,停止对所述当前的调度函数的子函数进行追踪;其中,所述局部安全函数包括第一类安全函数和/或第二类安全函数;所述第一类安全函数为不能触及所述污点汇聚函数的函数,所述第二类安全函数为输入参数不会影响到所述污点汇聚函数的函数;

当所述当前的调度函数未被标记为局部安全函数,且所述当前的调度函数不是所述污点汇聚函数时,继续对所述当前的调度函数的子函数进行追踪,直到所述当前的调度函数的最后一个子函数仍不是所述污点汇聚函数时,将所述当前的调度函数标记为所述第一类安全函数,并进行剪枝;

当所述当前的调度函数未被标记为局部安全函数,且所述当前的调用函数不是所述污点汇聚函数时,继续对所述当前的调度函数的子函数进行追踪,直到所述当前的调度函数的最后一个子函数为所述污点汇聚函数时,若所述当前的调度函数的输入参数不影响所述污点汇聚函数,将所述当前的调度函数标记为所述第二类安全函数,并进行剪枝。

[0070] 需要指出的是,尽管上述实施例中将各个步骤按照特定的先后顺序进行了描述,但是本领域技术人员可以理解,为了实现本申请的效果,不同的步骤之间并非必须按照这样的顺序执行,其可以同时(并行)执行或以其他顺序执行,这些变化都在本申请的保护范围之内。

[0071] 本领域技术人员能够理解的是,本申请实现上述一实施例的方法中的全部或部分流程,也可以通过计算机程序来指令相关的硬件来完成,所述的计算机程序可存储于一计算机可读存储介质中,该计算机程序在被处理器执行时,可实现上述各个方法实施例的步骤。其中,所述计算机程序包括计算机程序代码,所述计算机程序代码可以为源代码形式、

对象代码形式、可执行文件或某些中间形式等。所述计算机可读存储介质可以包括:能够携带所述计算机程序代码的任何实体或装置、介质、U盘、移动硬盘、磁碟、光盘、计算机存储器、只读存储器、随机存取存储器、电载波信号、电信信号以及软件分发介质等。需要说明的是,所述计算机可读存储介质包含的内容可以根据司法管辖区内立法和专利实践的要求进行适当的增减,例如在某些司法管辖区,根据立法和专利实践,计算机可读存储介质不包括电载波信号和电信信号。

[0072] 进一步,本申请还提供了一种固件漏洞的检测设备。

[0073] 参阅附图6,图6是根据本申请的一个实施例的固件漏洞的检测设备的主要结构框图。如图6所示,本申请实施例中的固件漏洞的检测设备可以包括处理器61和存储装置62。

[0074] 存储装置62可以被配置成存储执行上述方法实施例的固件漏洞的检测方法的程序,处理器61可以被配置成用于执行存储装置62中的程序,该程序包括但不限于执行上述方法实施例的固件漏洞的检测方法的程序。为了便于说明,仅示出了与本申请实施例相关的部分,具体技术细节未揭示的,请参照本申请实施例方法部分。该固件漏洞的检测设备可以是包括各种电子设备形成的控制设备。

[0075] 在一个具体实现过程中,该存储装置62和处理器61的数目均可以为多个。而执行上述方法实施例的固件漏洞的检测方法的程序可以被分割成多段子程序,每段子程序分别可以由处理器61加载并运行以执行上述方法实施例的固件漏洞的检测方法的不同步骤。具体地,每段子程序可以分别存储在不同的存储装置62中,每个处理器61可以被配置成用于执行一个或多个存储装置62中的程序,以共同实现上述方法实施例的固件漏洞的检测方法,即每个处理器61分别执行上述方法实施例的固件漏洞的检测方法的不同步骤,来共同实现上述方法实施例的固件漏洞的检测方法。

[0076] 上述多个处理器61可以是部署于同一个设备上的处理器,例如上述设备可以是由多个处理器组成的高性能设备,上述多个处理器61可以是该高性能设备上配置的处理器。此外,上述多个处理器61也可以是部署于不同设备上的处理器,例如上述设备可以是服务器集群,上述多个处理器61可以是服务器集群中不同服务器上的处理器。

[0077] 进一步,本申请还提供了一种计算机可读存储介质。在根据本申请的一个计算机可读存储介质实施例中,计算机可读存储介质可以被配置成存储执行上述方法实施例的智能家居设备的控制方法的程序,该程序可以由处理器加载并运行以实现上述固件漏洞的检测方法。为了便于说明,仅示出了与本申请实施例相关的部分,具体技术细节未揭示的,请参照本申请实施例方法部分。该计算机可读存储介质可以是包括各种电子设备形成的存储装置设备,可选的,本申请实施例中计算机可读存储介质是非暂时性的计算机可读存储介质。

[0078] 进一步,应该理解的是,由于各个模块的设定仅仅是为了说明本申请的装置的功能单元,这些模块对应的物理器件可以是处理器本身,或者处理器中软件的一部分,硬件的一部分,或者软件和硬件结合的一部分。因此,图中的各个模块的数量仅仅是示意性的。

[0079] 本领域技术人员能够理解的是,可以对装置中的各个模块进行适应性地拆分或合并。对具体模块的这种拆分或合并并不会导致技术方案偏离本申请的原理,因此,拆分或合并之后的技术方案都将落入本申请的保护范围内。

[0080] 需要说明的是,本申请各实施例中可能涉及的相关用户个人信息,均为严格按照

法律法规的要求,遵循合法、正当、必要的原则,基于业务场景的合理目的,处理用户在使用产品/服务过程中主动提供或因使用产品/服务而产生的,以及经用户授权获取的个人信息。

[0081] 本申请处理的用户个人信息会因具体产品/服务场景而有所不同,需以用户使用产品/服务的具体场景为准,可能会涉及用户的账号信息、设备信息、行驶信息、车辆信息或其他相关信息。本申请会以高度的勤勉义务对待用户的个人信息及其处理。

[0082] 本申请非常重视用户个人信息的安全,已采取符合业界标准、合理可行的安全防护措施保护用户的信息,防止个人信息遭到未经授权访问、公开披露、使用、修改、损坏或丢失。

[0083] 至此,已经结合附图所示的优选实施方式描述了本申请的技术方案,但是,本领域技术人员容易理解的是,本申请的保护范围显然不局限于这些具体实施方式。在不偏离本申请的原理的前提下,本领域技术人员可以对相关技术特征作出等同的更改或替换,这些更改或替换之后的技术方案都将落入本申请的保护范围之内。

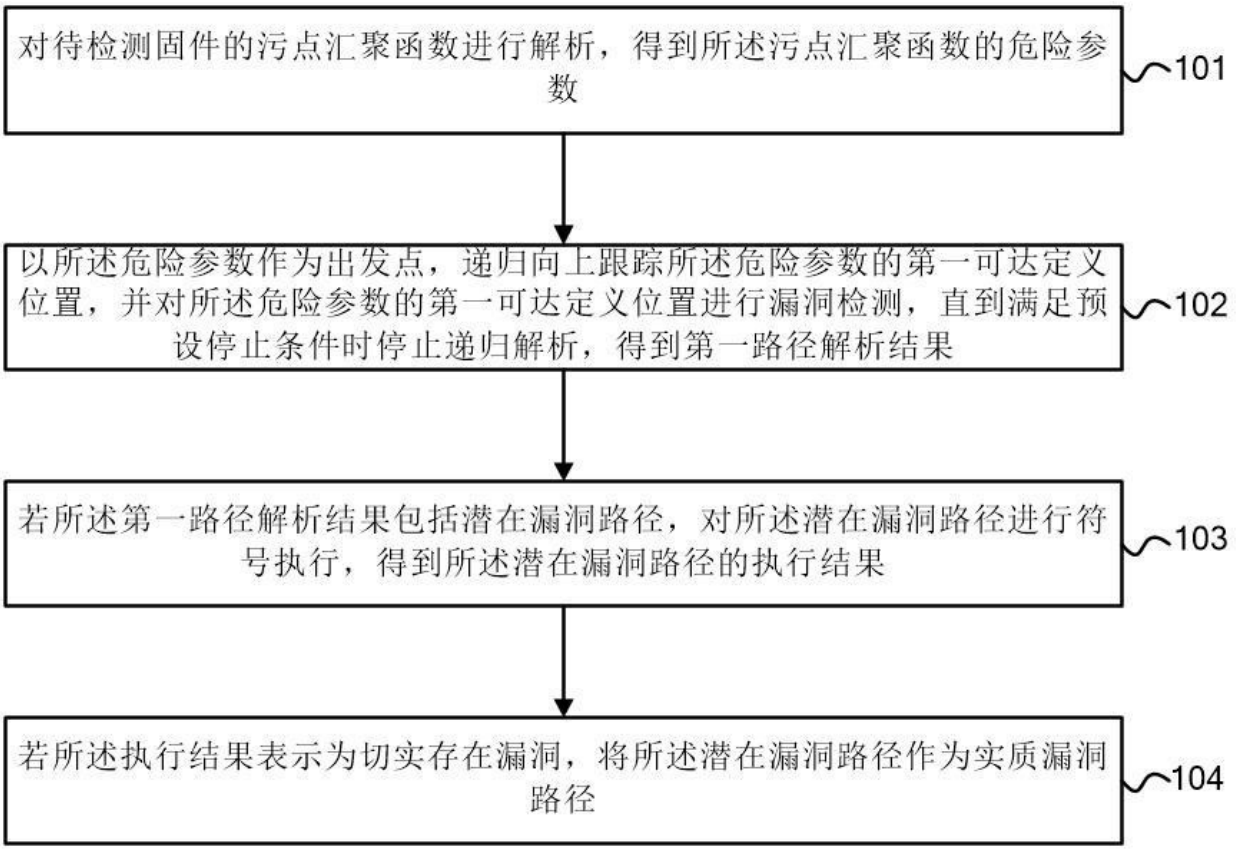

对待检测固件的污点汇聚函数进行解析，得到所述污点汇聚函数的危险参数
101
以所述危险参数作为出发点，递归向上跟踪所述危险参数的第一可达定义位置，并对所述危险参数的第一可达定义位置进行漏洞检测，直到满足预设停止条件时停止递归解析，得到第一路径解析结果
102
若所述第一路径解析结果包括潜在漏洞路径，对所述潜在漏洞路径进行符号执行，得到所述潜在漏洞路径的执行结果
103
若所述执行结果表示为切实存在漏洞，将所述潜在漏洞路径作为实质漏洞路径
104

图 1

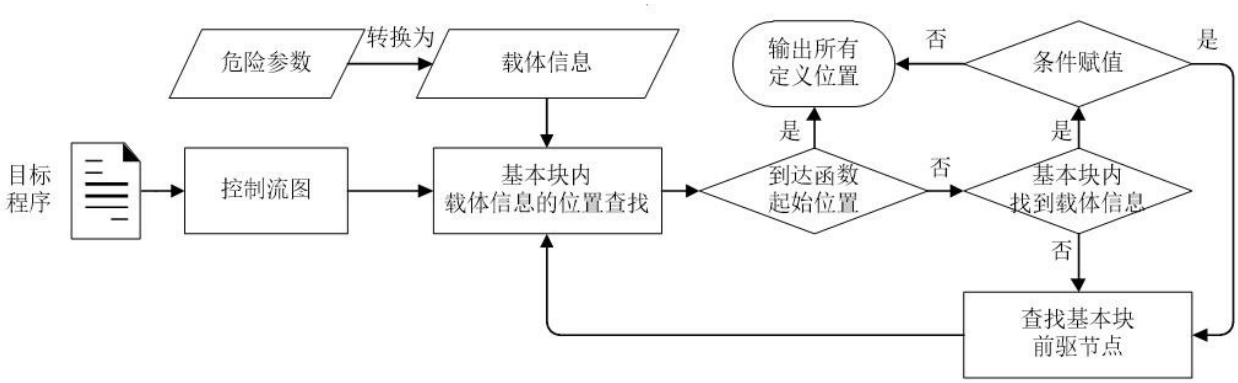

危险参数
转换为
载体信息
输出所有定义位置
否
条件赋值
是
目标程序
控制流图
基本块内载体信息的位置查找
是
到达函数起始位置
否
是
基本块内找到载体信息
否
查找基本块前驱节点

图 2

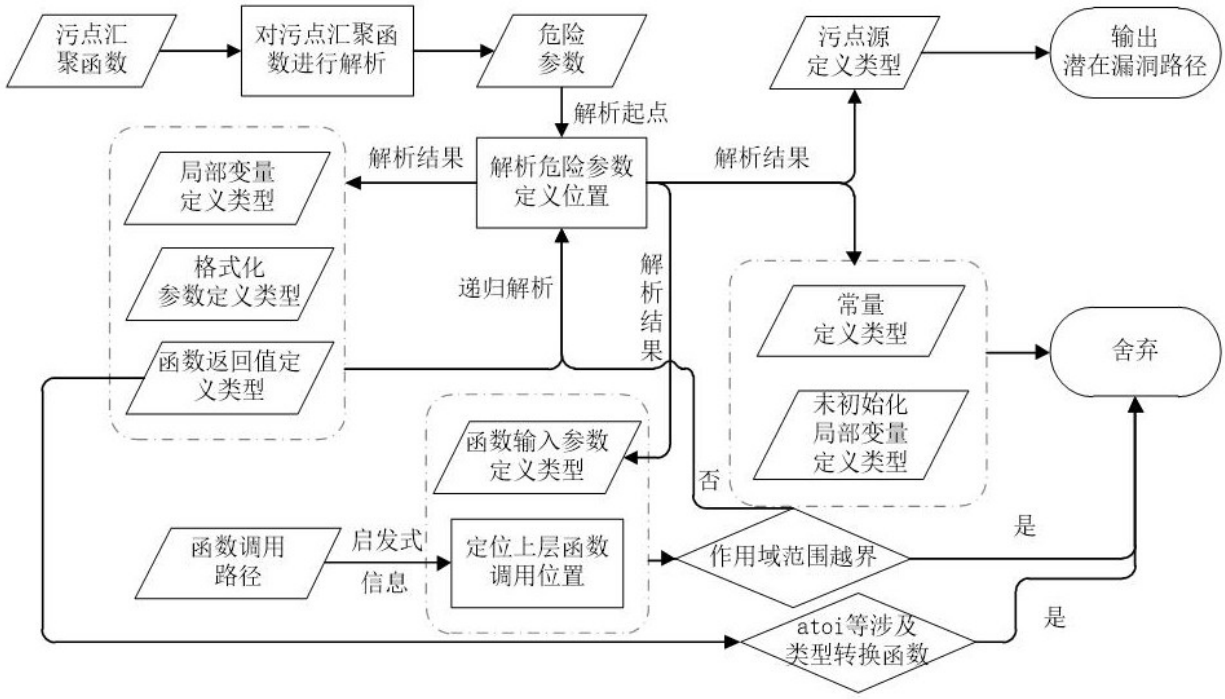

污点汇聚函数
对污点汇聚函数进行解析
危险参数
污点源定义类型
输出潜在漏洞路径
解析起点
局部变量定义类型
格式化参数定义类型
函数返回值定义类型
解析结果
解析危险参数定义位置
解析结果
递归解析
解析结果
常量定义类型
未初始化局部变量定义类型
舍弃
函数输入参数定义类型
否
函数调用路径
启发式信息
定位上层函数调用位置
作用域范围越界
是
atoi等涉及类型转换函数
是

图 3

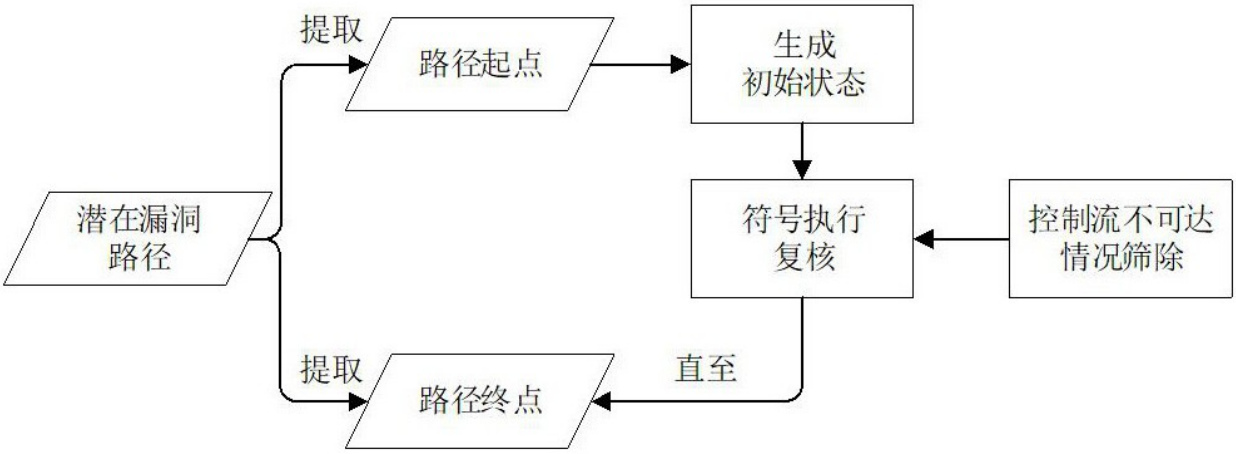

提取
路径起点
生成初始状态
潜在漏洞路径
符号执行复核
控制流不可达情况筛除
提取
路径终点
直至

图 4

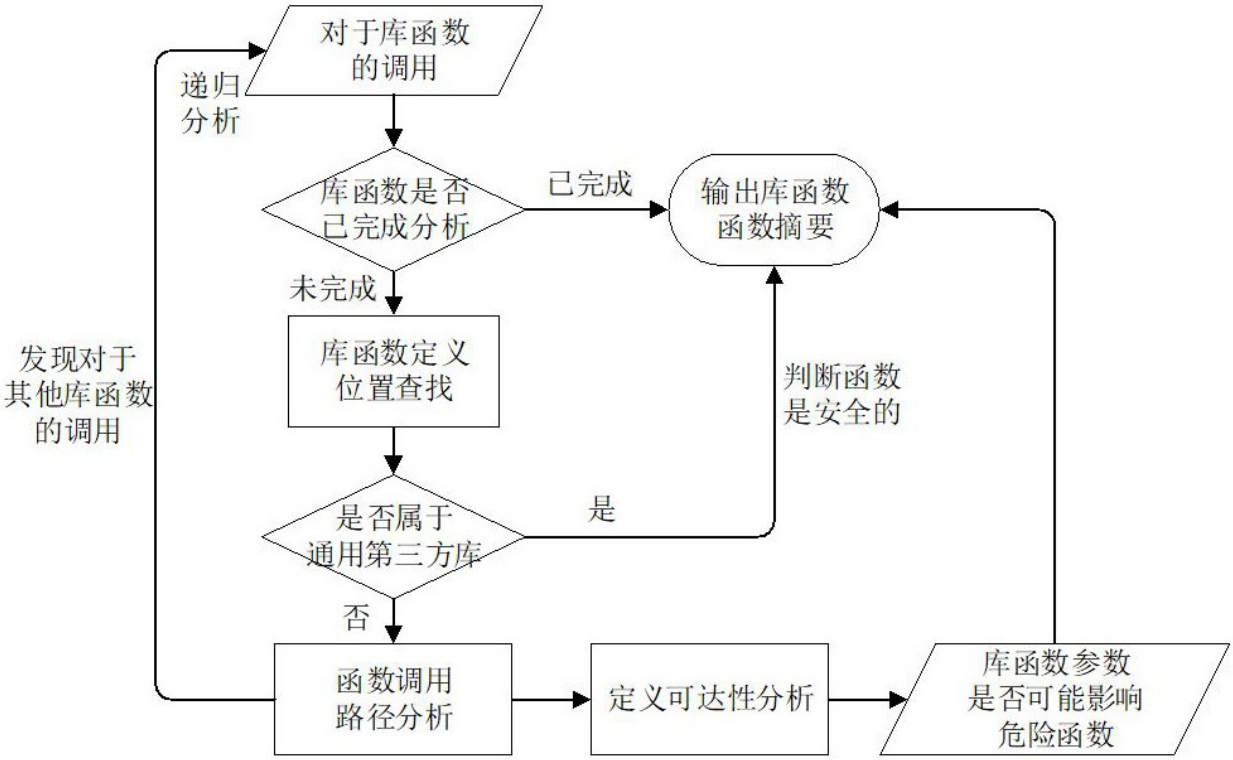

对于库函数
的调用
递归
分析
库函数是否
已完成分析
已完成
输出库函数
函数摘要
未完成
库函数定义
位置查找
判断函数
是安全的
发现对于
其他库函数
的调用
是否属于
通用第三方库
是
否
函数调用
路径分析
定义可达性分析
库函数参数
是否可能影响
危险函数

图 5

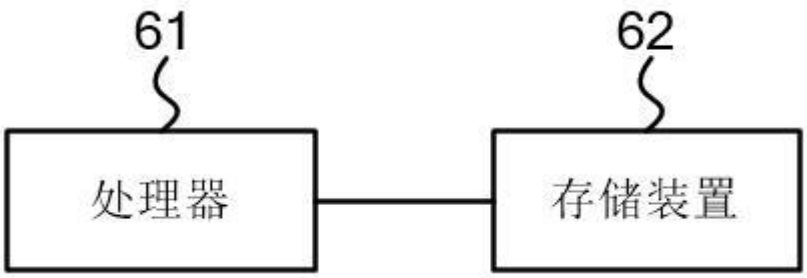

61
62
处理器
存储装置

图 6