(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 119473396 A
(43)申请公布日 2025.02.18

(21)申请号 202310954429.3
(22)申请日 2023.07.31
(71)申请人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼
申请人 南方科技大学
(72)发明人 何博远 请求不公布姓名 吴锦庭 童晓梦
(74)专利代理机构 深圳中一联合知识产权代理有限公司 44414
专利代理师 贾敏
(51)Int.Cl.
G06F 9/38(2018.01)
G06F 9/30(2018.01)
G06F 21/62(2013.01)
G06F 21/57(2013.01)

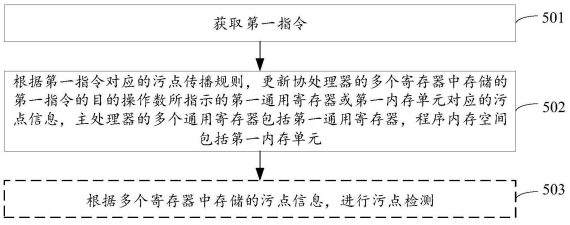
权利要求书3页 说明书20页 附图4页

(54)发明名称

污点分析方法、装置、计算机设备及存储介质

(57)摘要

本申请实施例公开了一种污点分析方法、装置、计算机设备及存储介质，属于信息安全技术领域。在本申请中，计算机设备的协处理器包括多个寄存器，该多个寄存器不仅可以用于存储计算机设备的主处理器中的多个通用寄存器对应的污点信息，还可以用于存储主处理器当前执行的目标程序的程序内存空间对应的污点信息。这样，协处理器在对第一指令进行污点传播时，可以通过访问该多个寄存器来访问第一指令的操作数的污点信息，相较于通过访问内存来访问污点信息，操作更为简单，速度更快，性能开销更小。



CN 119473396 A

1.一种污点分析方法,其特征在于,应用于计算机设备中的协处理器,所述协处理器包括多个寄存器,所述多个寄存器用于存储所述计算机设备的主处理器中的多个通用寄存器对应的污点信息,以及所述主处理器当前执行的目标程序的程序内存空间对应的污点信息,所述方法包括:

获取第一指令,所述第一指令为所述目标程序中的指令;

根据所述第一指令对应的污点传播规则,更新所述第一指令的目的操作数所指示的第一通用寄存器或第一内存单元对应的污点信息,所述多个通用寄存器包括所述第一通用寄存器,所述程序内存空间包括所述第一内存单元。

2.根据权利要求1所述的方法,其特征在于,所述程序内存空间包括栈区,所述程序内存空间对应的污点信息包括所述栈区中的内存单元对应的污点信息,所述栈区中的内存单元对应的污点信息包括污点标签和信息类型标签,所述内存单元对应的污点标签用于指示所述内存单元所存储的信息是否被污染,所述内存单元对应的信息类型标签用于指示所述内存单元所存储的信息为数据或堆地址。

3.根据权利要求2所述的方法,其特征在于,所述程序内存空间还包括全局区,所述程序内存空间对应的污点信息还包括所述全局区中的内存单元对应的污点标签。

4.根据权利要求1至3任一所述的方法,其特征在于,所述通用寄存器对应的污点信息包括污点标签和信息类型标签,所述通用寄存器对应的污点标签用于指示所述通用寄存器所存储的信息是否被污染,所述通用寄存器对应的信息类型标签用于指示所述通用寄存器所存储的信息为数据或堆地址,当所述通用寄存器对应的信息类型标签指示所述通用寄存器所存储的信息为堆地址时,所述通用寄存器对应的污点信息还包括所述通用寄存器存储的堆地址在所述程序内存空间中的存储地址。

5.根据权利要求1至4任一所述的方法,其特征在于,所述获取第一指令,包括:

从指令缓存队列中读取所述第一指令,所述指令缓存队列用于存储已解码的多个指令,所述多个指令为所述目标程序中所述主处理器已执行的指令。

6.根据权利要求5所述的方法,其特征在于,所述方法还包括:

从指令信息队列中读取第一指令信息,所述指令信息队列用于存储所述主处理器发送的多个指令信息,所述第一指令信息包括未解码指令;

对所述第一指令信息包括的未解码指令进行解码,得到所述第一指令,并将所述第一指令存储至所述指令缓存队列。

7.根据权利要求6所述的方法,其特征在于,所述方法还包括:

若所述指令信息队列已满,则向所述主处理器发送中断指令,所述中断指令用于指示所述主处理器中断发送指令信息。

8.根据权利要求1至7任一所述的方法,其特征在于,所述方法还包括:

根据所述多个寄存器中存储的污点信息,进行污点检测。

9.根据权利要求8所述的方法,其特征在于,所述根据所述多个寄存器中存储的污点信息,进行污点检测,包括:

获取第二指令,所述第二指令为所述目标程序中的指令;

若所述第二指令为敏感指令且所述第二指令的操作数所指示的通用寄存器或内存单元对应的污点信息指示所存储的信息被污染,则向所述主处理器发送中断指令,并在指定

内存区域内写入所述第二指令的程序计数值。

10.根据权利要求1至9任一所述的方法,其特征在于,所述主处理器和所述协处理器为基于精简指令集计算机RISC-V架构的处理器。

11.一种污点分析装置,其特征在于,部署于计算机设备中的协处理器,所述协处理器包括多个寄存器,所述多个寄存器用于存储所述计算机设备的主处理器中的多个通用寄存器对应的污点信息,以及所述主处理器当前执行的目标程序的程序内存空间对应的污点信息,所述装置包括:

指令获取模块,用于获取第一指令,所述第一指令为所述目标程序中的指令;

污点传播模块,用于根据所述第一指令对应的污点传播规则,更新所述第一指令的目的操作数所指示的第一通用寄存器或第一内存单元对应的污点信息,所述多个通用寄存器包括所述第一通用寄存器,所述程序内存空间包括所述第一内存单元。

12.根据权利要求11所述的装置,其特征在于,所述程序内存空间包括栈区,所述程序内存空间对应的污点信息包括所述栈区中的内存单元对应的污点信息,所述栈区中的内存单元对应的污点信息包括污点标签和信息类型标签,所述内存单元对应的污点标签用于指示所述内存单元所存储的信息是否被污染,所述内存单元对应的信息类型标签用于指示所述内存单元所存储的信息为数据或堆地址。

13.根据权利要求12所述的装置,其特征在于,所述程序内存空间还包括全局区,所述程序内存空间对应的污点信息还包括所述全局区中的内存单元对应的污点标签。

14.根据权利要求11至13任一所述的装置,其特征在于,所述通用寄存器对应的污点信息包括污点标签和信息类型标签,所述通用寄存器对应的污点标签用于指示所述通用寄存器所存储的信息是否被污染,所述通用寄存器对应的信息类型标签用于指示所述通用寄存器所存储的信息为数据或堆地址,当所述通用寄存器对应的信息类型标签指示所述通用寄存器所存储的信息为堆地址时,所述通用寄存器对应的污点信息还包括所述通用寄存器存储的堆地址在所述程序内存空间中的存储地址。

15.根据权利要求11至14任一所述的装置,其特征在于,所述指令获取模块具体用于:

从指令缓存队列中读取所述第一指令,所述指令缓存队列用于存储已解码的多个指令,所述多个指令为所述目标程序中所述主处理器已执行的指令。

16.根据权利要求15所述的装置,其特征在于,所述指令获取模块还用于:

从指令信息队列中读取第一指令信息,所述指令信息队列用于存储所述主处理器发送的多个指令信息,所述第一指令信息包括未解码指令;

对所述第一指令信息包括的未解码指令进行解码,得到所述第一指令,并将所述第一指令存储至所述指令缓存队列。

17.根据权利要求16所述的装置,其特征在于,所述装置还用于:

若所述指令信息队列已满,则向所述主处理器发送中断指令,所述中断指令用于指示所述主处理器中断发送指令信息。

18.根据权利要求11至17任一所述的装置,其特征在于,所述装置还包括:

污点检测模块,用于根据所述多个寄存器中存储的污点信息,进行污点检测。

19.根据权利要求18所述的装置,其特征在于,

所述指令获取模块,还用于获取第二指令,所述第二指令为所述目标程序中的指令;

所述污点检测模块,具体用于若所述第二指令为敏感指令且所述第二指令的操作数所指示的通用寄存器或内存单元对应的污点信息指示所存储的信息被污染,则向所述主处理器发送中断指令,并在指定内存区域内写入所述第二指令的程序计数值。

20.根据权利要求11至19任一所述的装置,其特征在于,所述主处理器和所述协处理器为基于精简指令集计算机RISC-V架构的处理器。

21.一种计算机设备,其特征在于,所述计算机设备包括主处理器和协处理器,所述主处理器用于执行目标程序,所述协处理器包括多个寄存器,所述多个寄存器用于存储所述计算机设备的主处理器中的多个通用寄存器对应的污点信息,以及所述主处理器当前执行的目标程序的程序内存空间对应的污点信息,所述协处理器用于执行内存中存储的至少一条程序指令或代码,以实现权利要求1至10任一项所述的污点分析方法。

22.一种计算机可读存储介质,其特征在于,所述计算机可读存储介质存储有指令,当所述指令在计算机设备的协处理器上运行时,使得所述计算机设备的协处理器执行权利要求1至10任一项所述的污点分析方法。

污点分析方法、装置、计算机设备及存储介质

技术领域

[0001] 本申请涉及信息安全技术领域,尤其涉及一种污点分析方法、装置、计算机设备及存储介质。

背景技术

[0002] 随着互联网、云计算和移动终端等技术的蓬勃发展,计算机系统中的信息安全问题越来越不容忽视。动态污点分析方法是目前被广泛应用于漏洞检测、隐私数据泄露检测等信息安全检测领域的一种信息流分析技术。该方法可以通过将敏感数据标记为污点数据并追踪污点数据在程序运行过程中的传播情况,来检测计算机系统中的信息安全问题。

[0003] 相关技术中,计算机设备中可以配置有主处理器和协处理器。主处理器用于执行待测的目标程序。其中,在主处理器执行目标程序的过程中,对于目标程序中的任一条指令时,协处理器可以读取该指令的源操作数对应的污点标签,并根据污点传播规则以及源操作数对应的污点标签来判定该指令的目的操作数是否被污染,进而根据判定结果对目的操作数对应的污点标签进行更新。其中,指令的任一操作数可以用于指示一个内存单元或主处理器中的一个通用寄存器,操作数对应的污点标签则用于指示操作数所指示的内存单元或通用寄存器所存储的信息是否被污染,并且,当操作数指示的是内存单元时,操作数对应的污点标签将存储在内存中。基于此,当指令的源操作数和/或目的操作数指示的是内存单元时,协处理器在进行污点传播的过程中,要通过访问内存来读取和/或更新污点标签,性能开销较大。

发明内容

[0004] 本申请提供一种污点分析方法、装置、计算机设备及存储介质,可以降低协处理器进行污点分析时的性能开销。

[0005] 为达到上述目的,本申请采用如下技术方案:

[0006] 第一方面,提供一种污点分析方法,应用于计算机设备中的协处理器,所述协处理器包括多个寄存器,所述多个寄存器用于存储所述计算机设备的主处理器中的多个通用寄存器对应的污点信息,以及所述主处理器当前执行的目标程序的程序内存空间对应的污点信息,所述方法包括:获取第一指令,所述第一指令为所述目标程序中的指令;根据所述第一指令对应的污点传播规则,更新所述第一指令的目的操作数所指示的第一通用寄存器或第一内存单元对应的污点信息,所述多个通用寄存器包括所述第一通用寄存器,所述程序内存空间包括所述第一内存单元。

[0007] 其中,第一指令为当前待分析的指令,且第一指令不为污点标记类指令也不为污点检测类指令。

[0008] 在本申请中,计算机设备的协处理器包括多个寄存器,该多个寄存器不仅可以用于存储计算机设备的主处理器中的多个通用寄存器对应的污点信息,还可以用于存储主处理器当前执行的目标程序的程序内存空间对应的污点信息。这样,协处理器在对第一指令

进行污点传播时,可以通过访问该多个寄存器来访问第一指令的操作数的污点信息,相较于通过访问内存来访问污点信息,操作更为简单,速度更快,性能开销更小。

[0009] 可选地,所述程序内存空间包括栈区,所述程序内存空间对应的污点信息包括所述栈区中的内存单元对应的污点信息,所述栈区中的内存单元对应的污点信息包括污点标签和信息类型标签,所述内存单元对应的污点标签用于指示所述内存单元所存储的信息是否被污染,所述内存单元对应的信息类型标签用于指示所述内存单元所存储的信息为数据或堆地址。

[0010] 在本申请中,栈区中的每个内存单元对应的污点信息不仅包括用于指示相应内存单元存储的信息是否被污染的污点标签,还包括用于指示相应内存单元存储的信息为数据或堆地址的信息类型标签。由于栈区中的堆地址可以为对象类型的变量在堆区中的起始地址,所以通过栈区中每个内存单元对应的污点信息不仅可以标记栈区中存储的数据的污染情况,还可以同时标记堆区中的对象类型的变量的污染情况。也即,本申请实施例通过栈区中的每个内存单元对应的污点信息可以实现针对栈区的内存单元粒度的污点标记,同时,还可以实现针对堆区的变量粒度的污点标记。

[0011] 可选地,所述程序内存空间还包括全局区,所述程序内存空间对应的污点信息还包括所述全局区中的内存单元对应的污点标签。

[0012] 在本申请中,通过全局区中的每个内存单元对应的污点信息可以实现针对全局区的基于内存单元这一粒度的污点标记。例如,当以一个字节为一个内存单元时,则通过全局区中的每个内存单元对应的污点信息可以实现针对全局区的字节粒度的污点标记。

[0013] 可选地,所述通用寄存器对应的污点信息包括污点标签和信息类型标签,所述通用寄存器对应的污点标签用于指示所述通用寄存器所存储的信息是否被污染,所述通用寄存器对应的信息类型标签用于指示所述通用寄存器所存储的信息为数据或堆地址,当所述通用寄存器对应的信息类型标签指示所述通用寄存器所存储的信息为堆地址时,所述通用寄存器对应的污点信息还包括所述通用寄存器存储的堆地址在所述程序内存空间中的存储地址。

[0014] 在本申请中,通用寄存器对应的污点信息不仅包括用于指示相应通用寄存器存储的信息是否被污染的污点标签,还包括用于指示相应通用寄存器存储的信息为数据或堆地址的信息类型标签。这样,通过通用寄存器对应的污点信息不仅可以实现针对通用寄存器中的数据的污点标记,同时,还可以实现针对堆区的变量的污点标记。

[0015] 可选地,所述获取第一指令的实现过程可以包括:从指令缓存队列中读取所述第一指令,所述指令缓存队列用于存储已解码的多个指令,所述多个指令为所述目标程序中所述主处理器已执行的指令。

[0016] 在本申请中,通过指令缓存队列可以存储多个已解码的指令,这样,使得协处理器获取指令信息并解码指令的操作与读取已解码的指令进行污点分析的操作解耦,以流水线的方式并行执行,从而提高了协处理器进行污点分析的效率。

[0017] 可选地,在从指令缓存队列中读取第一指令之前,所述方法还包括:从指令信息队列中读取第一指令信息,所述指令信息队列用于存储所述主处理器发送的多个指令信息,所述第一指令信息包括未解码指令;对所述第一指令信息包括的未解码指令进行解码,得到所述第一指令,并将所述第一指令存储至所述指令缓存队列。

[0018] 在本申请中,可以通过指令信息队列缓存主处理器提交的多个指令信息,这样,协处理器可以并行执行接收指令信息与读取并处理指令信息的操作,提高了处理效率。并且,相较于主处理器每提交一个指令信息,等待协处理器分析完成后才执行下一个指令以提交下一个指令信息的方法,能够有效的降低协处理器阻塞主处理器所带来的性能开销。

[0019] 可选地,所述方法还包括:若所述指令信息队列已满,则向所述主处理器发送中断指令,所述中断指令用于指示所述主处理器中断发送指令信息。

[0020] 可选地,所述方法还包括:根据所述多个寄存器中存储的污点信息,进行污点检测。

[0021] 可选地,所述根据所述多个寄存器中存储的污点信息,进行污点检测的实现过程可以包括:获取第二指令,所述第二指令为所述目标程序中的指令;若所述第二指令为敏感指令且所述第二指令的操作数所指示的通用寄存器或内存单元对应的污点信息指示所存储的信息被污染,则向所述主处理器发送中断指令,并在指定内存区域内写入所述第二指令的程序计数值。

[0022] 在本申请中,协处理器在检测到敏感指令之后,通过读取多个寄存器中存储的污点信息即可以实现污点检测,而无需通过访问内存来读取污点信息,提高了污点检测的速度,减小了性能开销。

[0023] 可选地,所述主处理器和所述协处理器为基于精简指令集计算机(reduced instruction set computer,RISC-V)架构的处理器。由于RSIC-V架构相较于其他的开源架构具有更高的开源程序,因此,在采用RSIC-V架构的主处理器和协处理器上运用上述污点分析方法,可以降低污点分析的成本。

[0024] 第二方面,提供一种污点分析装置,所述污点分析装置包括至少一个模块,所述至少一个模块用于执行上述第一方面所述的污点分析方法。

[0025] 第三方面,提供一种计算机设备,所述计算机设备包括主处理器和协处理器,所述主处理器用于执行目标程序,所述协处理器包括多个寄存器,所述多个寄存器用于存储所述计算机设备的主处理器中的多个通用寄存器对应的污点信息,以及所述主处理器当前执行的所述目标程序的程序内存空间对应的污点信息,所述协处理器用于执行内存中存储的至少一条程序指令或代码,以实现上述第一方面所述的污点分析方法。

[0026] 第四方面,提供一种计算机可读存储介质,所述计算机可读存储介质中存储有指令,当所述指令在计算机设备的协处理器上运行时,使得所述计算机设备的协处理器执行上述第一方面所述的污点分析方法。

[0027] 第五方面,提供了一种包含指令的计算机程序产品,当该计算机程序产品在计算机设备上运行时,使得计算机设备的协处理器执行上述第一方面所述的污点分析方法。

[0028] 上述第二方面、第三方面、第四方面和第五方面所获得的技术效果与第一方面中对应的技术手段获得的技术效果近似,在这里不再赘述。

[0029] 在本申请实施例中,计算机设备的协处理器包括多个寄存器,该多个寄存器不仅可以用于存储计算机设备的主处理器中的多个通用寄存器对应的污点信息,还可以用于存储主处理器当前执行的目标程序的程序内存空间对应的污点信息。这样,协处理器在对第一指令进行污点传播时,可以通过访问该多个寄存器来访问第一指令的操作数的污点信息,相较于通过访问内存来访问污点信息,操作更为简单,速度更快,性能开销更小。

附图说明

[0030] 图1为本申请实施例提供的一种计算机设备的结构示意图；

[0031] 图2为本申请实施例提供的一种程序内存空间与对应的污点信息的示意图；

[0032] 图3为本申请实施例提供的另一种主处理器的程序内存空间与对应的污点信息的示意图；

[0033] 图4为本申请实施例提供的一种主处理器中的通用寄存器与对应的污点信息的示意图；

[0034] 图5为本申请实施例提供的一种污点分析方法的流程图；

[0035] 图6为本申请实施例提供的另一种污点分析方法的流程图；

[0036] 图7为本申请实施例提供的一种污点分析装置的结构示意图。

具体实施方式

[0037] 为使本申请实施例的目的、技术方案和优点更加清楚，下面将结合附图对本申请实施方式作进一步地详细描述。

[0038] 在对本申请实施例进行详细的解释说明之前，先对本申请实施例涉及的应用场景进行介绍。

[0039] 随着互联网、云计算和移动智能终端等技术的蓬勃发展，计算机系统对信息安全的需求越来越高。计算机系统信息安全的两个重要特性是信息的保密性和完整性。其中，信息的保密性是指系统中的数据在使用过程中不会发生泄露，完整性是指系统的数据在使用过程中不会被恶意的篡改或删除。

[0040] 目前，随着互联网技术的发展，在线支付、社交网络等互联网活动中涉及大量的用户数据。而第三方应用程序的使用导致用户数据难以被有效的保护。例如，计算机设备中安装的社交软件可能会由于存在安全漏洞而遭受攻击，导致用户的个人信息泄露。再例如，计算机设备所使用的在线支付网站也可能存在安全漏洞而遭受攻击，导致用户的账户信息和交易信息被盗取。由此可见，计算机系统的信息安全问题越来越不容忽视。基于此，用于漏洞挖掘、恶意代码检测以及敏感数据泄露检测等安全业务领域的动态污点分析技术成为了当下的研究热点之一。

[0041] 动态污点分析技术可以通过标记目标程序中的敏感数据并在目标程序的运行过程中追踪被标记数据的传播情况，来检测目标程序中存在的信息安全问题。目前，常用的动态污点分析技术有基于软件的动态污点分析方法以及硬件辅助的动态污点分析方法。其中，基于软件的动态污点分析方法由于需要通过动态二进制插桩的方法向目标程序的二进制文件中插入大量的分析代码，或者是需要通过虚拟机来执行目标程序以获得程序的执行信息，因此往往具有较大的性能开销。在此基础上，硬件辅助的动态污点分析方法越来越受到关注。本申请实施例即提供了一种基于协处理器的硬件辅助动态污点分析方法，可以应用于漏洞挖掘、恶意代码检测、敏感数据泄露检测等安全业务领域，以提高计算机系统的安全性。

[0042] 接下来对本申请实施例提供的污点分析方法所涉及的计算机设备的结构进行介绍。

[0043] 图1是本申请实施例提供的一种计算机设备的结构示意图。如图1所示，该计算机

设备可以包括主处理器101、协处理器102、通信总线103以及内存104。需要说明的是,图1示出的设备结构并不构成对计算机设备的限定,计算机设备可以包括比图示更多或更少的部件,或者组合某些部件,或者不同的部件布置,本申请实施例对此不进行限定。下面结合图1对计算机设备的各个构成部件进行详细的介绍:

[0044] 主处理器101是该计算机设备的控制中心,可以是一个处理器,也可以是多个处理元件的统称。例如,主处理器101可以是一个通用中央处理器(Central Processing Unit,CPU),或特定应用集成电路(application-specific integrated circuit,ASIC),或一个或多个用于控制本申请方案程序执行的集成电路,例如:一个或多个微处理器(digital signal processor,DSP),或,一个或者多个现场可编程门阵列(field programmable gate array,FPGA)。其中,主处理器101可以通过运行或执行存储在内存104内的软件程序,以及调用存储在内存104内的数据,执行计算机设备的各种功能。例如,主处理器101可以通过调用内存104中存储的插桩后的目标程序的二进制文件来执行目标程序,并将目标程序中的指令发送至协处理器102,以便协处理器102进行动态污点分析。

[0045] 协处理器102可以是一个用于对主处理器101执行的目标程序进行动态污点分析的硬件加速器。该协处理器102可以包括通用寄存器,除此之外,该协处理器102还可以包括多个用于存储污点标签的寄存器。其中,该多个寄存器不仅可以存储主处理器101中的通用寄存器对应的污点标签,还可以用于存储主处理器101中目标程序的程序内存空间对应的污点标签。另外,协处理器102可以访问内存104。例如,可以通过运行或执行存储在内存104中的软件程序来实现图5和图6所示的污点分析访问。

[0046] 通信总线103可包括一通路,在上述组件之间传送信息。该通信总线103可以是工业标准体系结构(industry standard architecture,ISA)总线、外部设备互连(peripheral component,PCI)总线或扩展工业标准体系结构(extended industry standard architecture,EISA)总线等。该总线可以分为地址总线、数据总线、控制总线等。为便于表示,图1中仅用一条粗线表示,但并不表示仅有一根总线或一种类型的总线。

[0047] 作为一种实施例,主处理器101和协处理器102可以为基于RISC-V架构的处理器。RISC-V架构是一种开源指令集架构,相较于其他的开源架构,RISC-V架构更加的短小而精悍、开源程度更高且指令集架构更加完成。并且,RISC-V架构支持自定义指令,支持自定义硬件加速器。例如,在本申请实施例中,主处理器101在通过通信总线103与协处理器102进行通信时,可以采用rocket定制协处理器(rocket custom coprocessor,RoCC)接口进行信息传输。

[0048] 作为一种实施例,计算机设备可以包括多个主处理器和多个协处理器,其中,每个主处理器对应至少一个协处理器。并且,多个主处理器中的每一个可以是一个单核(single-CPU)处理器,也可以是一个多核(multi-CPU)处理器。

[0049] 内存104是指与主处理器101直接交换数据的内部存储器,它可以随时读写数据,而且速度很快,作为操作系统或其他正在运行中的程序的临时数据存储器。内存包括至少两种存储器,例如内存既可以是随机存取存储器(random access memory,RAM),也可以是只读存储器(read only memory,ROM)。举例来说,RAM可以是动态随机存取存储器(dynamic random access memory,DRAM),或者存储级存储器(storage class memory,SCM)。DRAM是一种半导体存储器,与大部分随机存取存储器一样,属于一种易失性存储器(volatile

memory)设备。SCM是一种同时结合传统储存装置与存储器特性的复合型储存技术,SCM能够提供比硬盘更快速的读写速度,但存取速度上比DRAM慢,在成本上也比DRAM更为便宜。然而,DRAM和SCM在本实施例中只是示例性的说明,内存还可以包括其他随机存取存储器,例如静态随机存取存储器(static random access memory,SRAM)等。而对于只读存储器,举例来说,可以是可编程只读存储器(programmable read only memory,PROM)、可抹除可编程只读存储器(erasable programmable read only memory,EPROM)等。另外,内存104还可以是双列直插式存储器模块或双线存储器模块(dual in-line memory module,DIMM),即由DRAM组成的模块。实际应用中,计算机设备中可配置多个内存104,以及不同类型的内存104。本实施例不对内存104的数量和类型进行限定。另外,内存104可以是独立存在,例如,可以通过通信总线103与主处理器101和协处理器102相连接。或者,内存104也可以和主处理器101集成在一起。例如,参见图1,内存104可以包括主处理器101和协处理器102共享的数据缓存1041和RAM 1042,其中,该数据缓存1041可以与主处理器101集成在一起,RAM 1042则通过通信总线103与主处理器101和协处理器102连接。其中,内存104用于存储执行本申请实施例提供的方案的软件程序,并由协处理器102来控制执行。

[0050] 除此之外,计算机设备还可以包括通信接口105。该通信接口105用于与其他设备或通信网络通信,如以太网,RAN,无线局域网(wireless local area networks,WLAN)等。通信接口105可以包括接收单元实现接收功能,以及发送单元实现发送功能。

[0051] 作为一种实施例,计算机设备还可以包括输出设备106和输入设备107。输出设备106和主处理器101通信,可以以多种方式来显示信息。例如,输出设备106可以是液晶显示器(liquid crystal display,LCD),发光二极管(light emitting diode,LED)显示设备,阴极射线管(cathode ray tube,CRT)显示设备,或投影仪(projector)等。输入设备107和主处理器101通信,可以以多种方式接收输入。例如,输入设备107可以是鼠标、键盘、触摸屏设备或传感设备等。

[0052] 上述的计算机设备可以是一个用户终端,也可以是一个服务器。其中,当该计算机设备为用户终端时,该计算机设备可以为智能手机、平板电脑、便携式电脑、台式机、可穿戴设备等,本申请实施例对此不作限定。

[0053] 接下来对动态污点分析的基本实现过程进行简要介绍。

[0054] 动态污点分析的处理过程可以包括三个阶段:污点标记、污点传播和污点检测。

[0055] 污点标记是指为引入目标程序的数据设置污点标签,以此来指示该数据是否为污点数据,并存储该污点标签。目前,当引入的数据位于通用寄存器中时,可以通过为该数据当前所在的通用寄存器设置污点标签来指示该数据是否为污点数据,其中,通用寄存器对应的污点标签可以存储在其他寄存器中。当引入的数据存储在内存中时,则可以通过为该数据所在的内存单元设置污点标签来指示该数据是否为污点数据。其中,内存单元对应的污点标签可以存储在与该内存单元存在映射关系的影子内存中。

[0056] 污点传播是指在执行目标程序的过程中,根据目标程序中待分析的指令的源操作数的污点标签和污点传播规则来判定该指令的目的操作数是否被污染,进而根据判定结果来更新目的操作数的污点标签。其中,目的操作数用于指示通用寄存器或内存单元。由此可见,污点传播实际上是在基于指令中源操作数的污点标签来确定目的操作数所指示的通用寄存器或内存单元中存储的信息是否被污染,进而以此来更新目的操作数所指示的通用寄

存器或内存单元所对应的污点标签。

[0057] 污点检测是指在执行目标程序的过程,检测目标程序中待分析的指令是否为敏感指令,并在待分析的指令为敏感指令的情况下,检测该敏感指令的操作数是否为被污染的操作数,也即,检测该敏感指令所要操作的通用寄存器或内存单元中存储的信息是否被污染。如果该敏感指令的操作数为被污染的操作数,则触发异常报警。

[0058] 目前,硬件辅助的动态污点分析方法包括基于协处理器的动态污点分析方法。在该方法中,计算机设备中可以配置有主处理器和协处理器。主处理器用于执行待检测的目标程序。其中,在主处理器执行目标程序的过程中,对于目标程序中的任一条指令,协处理器可以读取该指令的源操作数对应的污点标签,并根据污点传播规则以及源操作数对应的污点标签来判定该指令的目的操作数是否被污染,进而根据判定结果对目的操作数对应的污点标签进行更新。其中,由前述介绍可知,操作数的污点标签实际上就是操作数所指示的通用寄存器或内存单元对应的污点标签,而通用寄存器对应的污点标签存储在其他寄存器中,内存单元对应的污点标签存储在影子内存中。基于此,当指令的源操作数和/或目的操作数指示的是内存单元时,协处理器在进行污点传播的过程中,可以通过访问内存来读取和/或更新污点标签。这样,协处理器在进行污点分析的过程中将会频繁的访问内存,性能开销较大。

[0059] 基于此,本申请实施例提供了一种基于协处理器的动态污点分析方法,在本申请实施例中,计算机设备的协处理器包括多个寄存器,该多个寄存器不仅可以用于存储计算机设备的主处理器中的多个通用寄存器对应的污点信息,还可以用于存储主处理器当前执行的目标程序的程序内存空间对应的污点信息。这样,协处理器在对第一指令进行污点传播时,可以通过访问该多个寄存器来访问第一指令的操作数的污点信息,相较于通过访问内存来访问污点信息,操作更为简单,速度更快,性能开销更小。

[0060] 接下来首先对本申请实施例提供的通过协处理器的多个寄存器存储污点信息的实现方式进行介绍。

[0061] 在本申请实施例中,协处理器除了可以包括通用寄存器之外,还可以包括多个用于存储污点信息的寄存器。其中,该多个寄存器不仅可以用于存储主处理器中的多个通用寄存器对应的污点信息,还可以用于存储主处理器当前执行的目标程序的程序内存空间对应的污点信息。

[0062] 示例性的,该多个寄存器中可以存储有影子寄存器文件和污点存储文件。其中,影子寄存器文件中包括主处理器的多个通用寄存器对应的污点信息,污点存储文件中包括目标程序的程序内存空间对应的污点信息。

[0063] 接下来首先对污点存储文件中的目标程序的程序内存空间对应的污点信息进行介绍。

[0064] 目标程序的程序内存空间也可以称为进程内存空间,是计算机设备在运行目标程序时为该目标程序分配的内存空间。该程序内存空间可以包括栈区、堆区、全局区和代码区。其中,栈区用于存储数据和堆地址,该数据可以是基本类型的变量,如整数、浮点数、字符串等。该堆地址可以是指对象类型的变量在堆区中的起始地址。栈区的空间大小是固定的,在本申请实施例中,栈区可以包括多个内存单元,其中,每个内存单元的大小可以相同。例如,可以以一个字节的内存空间为一个内存单元。堆区是目标程序运行过程中动态分配

的内存空间,因此,堆区的空间大小是不确定的。堆区可以用于存储对象类型的变量。全局区是用于存放全局变量和静态变量的内存区域,全局区可以包括未初始化数据段和初始化数据段。其中,未初始化数据段也可以称为BSS(block started by symbol)段,用于存储程序中未初始化的或者是初始值为0的全局变量和/或静态变量。初始化数据段也可以称为数据(data)段,用于存储程序中已初始化的全局变量和/或或静态变量。全局区也可以包括多个内存单元,其中,每个内存单元的大小可以相同,例如,在本申请实施例中,与栈区相同,也可以将一个字节的内存空间作为一个内存单元。代码区用于存储程序的可执行代码。

[0065] 基于上述介绍,在一种可能的实现方式中,目标程序的程序内存空间对应的污点信息可以包括栈区中的内存单元的污点信息。其中,栈区中的内存单元对应的污点信息可以包括污点标签和信息类型标签,该污点标签用于指示内存单元所存储的信息是否被污染,信息类型标签用于指示内存单元所存储的信息为数据或堆地址。

[0066] 示例性的,栈区中的每个内存单元对应的污点信息可以包括两个比特。其中,高位比特可以为污点标签,当高位比特为0时,用于指示该内存单元存储的信息未被污染,当高位比特为1时,用于指示该内存单元存储的信息已被污染。低位比特可以为信息类型标签,当低位比特为0时,用于指示该内存单元存储的信息为数据,当低位比特为1时,用于指示该内存单元存储的信息为堆地址。需要说明的是,当低位比特为1时,由于该内存单元存储的信息为堆地址,所以,高位比特指示该内存单元存储的信息是否被污染,实际上就是指示该内存单元所存储的堆地址是否被污染,也即,指示该内存单元所存储的堆地址上存储的数据是否被污染。由此可见,本申请实施例中通过栈区中的内存单元对应的污点信息可以间接的指示堆区的污染情况。

[0067] 在本申请实施例中,栈区中的每个内存单元对应的污点信息可以按照内存单元的地址从高到低的顺序依次存储在污点存储文件中。需要说明的是,由于一个寄存器可以存储的数据量有限,所以在本申请实施例中,可以通过多个寄存器中的不止一个寄存器来存储污点存储文件。

[0068] 可选地,污点存储文件可以包括第一子文件,栈区中的每个内存单元对应的污点信息可以按照内存单元的地址从高到底的顺序依次存储在该第一子文件中。

[0069] 例如,参见图2,栈区中地址为0x0fffff9f8的内存单元中存放有一个堆地址0x000863e8,堆区中堆地址为0x000863e8的区域存储有常量字符串“BBB”,假设该常量字符串“BBB”为未被污染的数据,则第一子文件中存储的栈区中地址为0x0fffff9f8的内存单元对应的污点信息为“01”,其中,“1”为该内存单元的信息类型标签,用于指示地址为0x0fffff9f8的内存单元中存储的是一个堆地址,“0”为该内存单元的污点标签,用于指示地址为0x0fffff9f8的内存单元中存储的堆地址0x000863e8中存储的数据为未被污染的数据。

[0070] 再例如,仍然参见图2,栈区中地址为0x0fffff9f0的内存单元中存放有一个局部变量数组“AAA”,假设该局部变量数组“AAA”已被污染,也即,是一个污点数据,则第一子文件中存储的地址为0x0fffff9f0的内存单元对应的污点信息为“10”。其中,“0”为该内存单元的信息类型标签,用于指示地址为0x0fffff9f0的内存单元中存储的是数据,“1”为该内存单元的污点标签,用于指示地址为0x0fffff9f0的内存单元中存储的数据“AAA”为被污染的数据,也即,污点数据。

[0071] 由此可见,在本申请实施例中,栈区中的每个内存单元对应的污点信息不仅包括用于指示相应内存单元存储的信息是否被污染的污点标签,还包括用于指示相应内存单元存储的信息为数据或堆地址的信息类型标签。由于栈区中的堆地址可以为对象类型的变量在堆区中的起始地址,所以通过栈区中每个内存单元对应的污点信息不仅可以标记栈区中存储的数据的污染情况,还可以同时标记堆区中的对象类型的变量的污染情况。也即,本申请实施例通过栈区中的每个内存单元对应的污点信息可以实现针对栈区的基于内存单元这一粒度的污点标记,同时,还可以实现针对堆区的基于变量粒度的污点标记。例如,当一个内存单元的大小为一个字节时,则通过栈区中每个内存单元对应的污点信息,可以实现针对栈区的字节粒度的污点标记,以及针对堆区的变量粒度的污点标记。

[0072] 可选地,协处理器的多个寄存器存储的目标程序的程序内存空间对应的污点信息还可以包括全局区中的内存单元对应的污点标签。

[0073] 需要说明的是,由于全局区用于存储全局变量和/或静态变量,而全局变量和静态变量均为值类型的变量,也即,全局区中不包括地址。所以,在本申请实施例中,全局区中的每个内存单元对应的污点信息可以包括污点标签而不包括信息类型标签。基于此,全局区中的每个内存单元对应的污点信息可以通过一个比特来表示,该比特即为相应内存单元对应的污点标签。示例性的,当该比特为1时,指示对应的内存单元中存储的数据被污染,也即对应的内存单元中存储的数据为污点数据。当该比特为0时,则指示对应的内存单元中存储的数据未被污染,不为污点数据。

[0074] 在本申请实施例中,全局区中每个内存单元对应的污点信息和栈区中的每个内存单元的污点信息可以按照各个内存单元的地址从高到低的顺序依次存储在污点存储文件中。

[0075] 可选地,如前所述,污点存储文件除了包括第一子文件,还可以包括第二子文件,在这种情况下,全局区中每个内存单元对应的污点信息可以按照内存单元的地址从高到低的顺序依次存储在第二子文件中。

[0076] 例如,参见图3,全局区中的内存单元M1中存储有全局变量数组“CCC”,假设该数组“CCC”为污点数据,则第二子文件中存储的该内存单元M1对应的污点信息可以为1。

[0077] 由此可见,在本申请实施例中,通过全局区中的每个内存单元对应的污点信息可以实现针对全局区的基于内存单元这一粒度的污点标记。例如,当以一个字节为一个内存单元时,则通过全局区中的每个内存单元对应的污点信息可以实现针对全局区的字节粒度的污点标记。

[0078] 上文中主要对污点存储文件进行了介绍,介绍对影子寄存器文件中存储的主处理器中的多个通用寄存器对应的污点信息进行介绍。

[0079] 通用寄存器可以是指用于传送或暂存数据的寄存器。示例性的,主处理器中的多个通用寄存器可以包括累加器、计数器、基址寄存器等。例如,当主处理器为RSIC-V架构的处理器时,该主处理器可以包括31个通用寄存器。

[0080] 在本申请实施例中,主处理器中的通用寄存器对应的污点信息可以包括污点标签和信息类型标签,其中,污点标签用于指示通用寄存器所存储的信息是否被污染,信息类型标签用于指示通用寄存器所存储的信息为数据或堆地址。

[0081] 可选地,当通用寄存器对应的信息类型标签指示通用寄存器所存储的信息为堆地

址时,通用寄存器对应的污点信息还可以包括通用寄存器存储的堆地址在程序内存空间中的存储地址。

[0082] 示例性的,在本申请实施例中,通用寄存器对应的污点信息可以包括N个比特,其中,N不小于M+2,M为内存地址的比特数,例如,当M等于16时,N可以等于32或64。基于此,N个比特中的最高位比特可以为污点标签。当该最高位比特为0时,也即,当污点标签为0时,用于指示对应的通用寄存器存储的信息未被污染;当最高位比特为1时,也即,当污点标签为1时,用于指示对应的通用寄存器存储的信息被污染。N个比特中的次高位比特可以为信息类型标签。当该次高位比特为0时,也即,当信息类型标签为0时,用于指示对应的通用寄存器存储的信息为数据,在这种情况下,N个比特中的剩余比特可以均为0;当次高位比特为1时,也即,当信息类型标签为1时,用于指示对应的通用寄存器存储的信息为堆地址,在这种情况下,N个比特中的剩余比特可以表示该堆地址在程序内存空间的栈区中的存储地址。另外,当通用寄存器中存储的信息为堆地址时,则该通用寄存器对应的污点标签指示该堆地址是否未被污染,实际上就是指示该堆地址中存储的数据是否未被污染。由此可见,在本申请实施例中,通过通用寄存器对应的污点信息不仅可以指示通用寄存器中的数据是否被污染,还可以指示堆区中的数据是否被污染。

[0083] 在本申请实施例中,各个通用寄存器对应的污点信息可以存储在影子寄存器文件中。需要说明的是,由于一个寄存器可以存储的数据量有限,所以在本申请实施例中,可以通过多个寄存器中的不止一个寄存器来存储影子寄存器文件。示例性的,主处理器中的每个通用寄存器可以映射至协处理器包括的多个寄存器中的一个寄存器,也即,每个通用寄存器与协处理器中用于存储污点信息的多个寄存器中的一个寄存器存在映射关系,这样,影子寄存器文件中每个通用寄存器对应的污点信息可以存储在与该通用寄存器存在映射关系的寄存器中。

[0084] 例如,参见图4,主处理器中的通用寄存器t0中存储有堆地址0x000863e8,假设该堆地址在栈区中的存储地址为0x0fffff9f8,且该堆地址上存储的数据未被污染,则在影子寄存器文件中该通用寄存器t0对应的污点信息为01 0x0fffff9f8,并且,该污点信息可以存储在与通用寄存器t0存在映射关系的寄存器t0’中。其中,该污点信息中的次高位比特“1”用于指示通用寄存器t0中存储的为一个堆地址,最高位比特“0”用于指示该通用寄存器t0中存储的堆地址中的数据未被污染,剩余的比特“0x0fffff9f8”则为通用寄存器t0中存储的堆地址在栈区中的存放地址。

[0085] 仍以图4为例,主处理器中的通用寄存器t1中存储有数据0x0000000a,假设该数据为污点数据,则在影子寄存器文件中该通用寄存器t1对应的污点信息可以为10 0x00000000,并且,该污点信息可以存储在与通用寄存器t1存在映射关系的寄存器t1’中。其中,该污点信息中的次高位比特“0”用于指示通用寄存器t1中存储的为一个数据,最高位比特“1”用于指示该通用寄存器t1中存储的数据被污染,为一个污点数据。由于通用寄存器t1中存储的是数据,所以剩余的比特为“0x00000000”。

[0086] 需要说明的是,上文中图2至图4所示出的堆地址、通用寄存器和内存单元中存储的数据以及内存单元的地址,是本申请实施例为了说明污点信息的实现方式给出的一种示例,并不构成对本申请实施例的限定。

[0087] 基于上述介绍的污点信息的实现方式,本申请实施例可以通过图5所示的污点分

析方法来进行动态污点分析,其中,该方法可以应用于计算机设备包括的协处理器中,参见图5,该方法包括以下步骤:

[0088] 步骤501:获取第一指令。

[0089] 在本申请实施例中,计算机设备中可以存储有插桩后的目标程序的可执行文件。其中,该可执行文件可以为二进制文件,并且,插桩后的目标程序的可执行文件中包括用于实现主处理器向协处理器发送指令信息以使协处理器进行污点分析的代码。主处理器可以执行该目标程序的可执行文件,以此来执行该目标程序。其中,主处理器每执行目标程序中的一条指令后,可以生成该指令的指令信息,并将该指令的指令信息发送至协处理器。其中,该指令的指令信息可以包括该指令的执行日志,指令的执行日志可以包括未解码的该指令,除此之外,指令的执行日志还可以包括该指令的程序计数值、该指令使用的内存单元和/或通用寄存器的地址以及该指令使用的数据。其中,指令的程序计数值用于指示该指令在内存中的存储位置。

[0090] 示例性的,当主处理器为RISC-V架构的处理器时,主处理器在生成该指令的指令信息之后,可以通过RoCC接口向协处理器发送该指令信息。协处理器在接收到该指令信息之后,可以将该指令信息中包括的未解码的指令进行解码,从而得到第一指令。

[0091] 其中,该第一指令可以包括操作码和操作数。操作码用于指示第一指令所要执行的操作,操作数用于指示第一指令所要操作的对象。其中,当第一指令为双操作数指令时,该第一指令的操作数可以包括目的操作数和源操作数,该目的操作数可以用于指示主处理器中的多个通用寄存器中的第一通用寄存器或目标程序的程序内存空间中的第一内存单元,该第一通用寄存器或第一内存单元即为第一指令操作后的数据的存放位置。该源操作数可以为立即数,此时,该立即数即为被操作的数据。或者,该源操作数也可以用于指示主处理器的多个通用寄存器中的第二通用寄存器或目标程序的程序内存空间中的第二内存单元。该第二通用寄存器或第二内存单元即为被操作的数据的存放位置。可选地,当第一指令为单操作数指令时,第一指令可以包括一个操作数,此时,该操作数既为源操作数也为目的操作数。

[0092] 可选地,在一种可能的实现方式中,协处理器中可以包括一个用于存储指令信息的指令信息队列。这样,协处理器在接收到主处理器发送的指令信息后,可以将该指令信息存储至该指令信息队列中。相应的,协处理器可以从该指令信息队列中读取第一指令信息,并对该第一指令信息包括的未解码指令进行解码,从而得到第一指令。

[0093] 需要说明的是,该指令信息队列可以为一个先进先出的队列。基于此,协处理器可以将接收到的主处理器发送的指令信息放入该指令信息队列的队尾,并从该指令信息队列的队首读取第一指令信息。这样,通过该指令信息队列可以缓存主处理器提交的多个指令信息,协处理器可以并行执行接收指令信息与读取并处理指令信息的操作,提高了处理效率。并且,相较于主处理器每提交一个指令信息,等待协处理器分析完成后才执行下一个指令以提交下一个指令信息的方法,能够有效的降低协处理器阻塞主处理器所带来的性能开销。

[0094] 可选地,目标程序中有可能会包含有与污点分析无关的指令,例如,某些不会造成污点传播的指令。基于此,协处理器还可以对接收到的主处理器发送的指令信息进行过滤,将与污点分析相关的指令的指令信息存入指令信息队列,而将与污点分析无关的指令的指

令信息丢弃,以此来减少协处理器所要分析的指令的数量,从而降低协处理器的负载和功耗。

[0095] 示例性的,在本申请实施例中,在对目标程序进行插桩时,可以预先针对目标程序包含的与污点分析无关的指令设置一组自定义的辅助过滤指令。其中,该辅助过滤指令可以包括开始过滤指令和停止过滤指令,可以将开始过滤指令插入至所要过滤的指令之前,将停止过滤指令插入至所要过滤的指令之后,也即,位于该开始过滤指令和停止过滤指令之间的指令即为要过滤掉的指令。主处理器在执行目标程序的过程中,检测到辅助过滤指令时,可以不执行该辅助过滤指令,而将该辅助过滤指令直接发送至协处理器。基于此,若协处理器接收到主处理器发送的辅助过滤指令中的开始过滤指令,则从该开始过滤指令起,将后续接收到的主处理器发送的各条指令信息丢弃,直至接收到停止过滤指令为止,继续将接收到的各条指令信息存放至指令信息队列。

[0096] 可选地,在一些可能的情况中,该指令信息队列能够存储的指令信息的数量是一定的。基于此,协处理器在将接收到的指令信息放入至指令信息队列之后,还可以检测该指令信息队列是否已满,若该指令信息队列已满,则协处理器可以向主处理器发送中断指令,以指示主处理器中断发送指令信息。

[0097] 示例性的,该指令信息队列能够存储预设条数的指令信息。基于此,协处理器可以检测当前指令信息队列中已存储的指令信息的条数,若已存储的指令信息的条数等于该预设条数,则该协处理器可以向主处理器发送中断指令。主处理器在接收到该中断指令之后,可以暂停执行目标程序的指令,以此来暂停向协处理器发送指令信息,并每隔预设时间间隔向协处理器查询指令信息队列中的指令信息的条数是否小于预设条数,若查询到指令信息队列中的指令信息的条数小于预设条数,则主处理器开始继续执行目标程序中的指令,并继续向协处理器发送执行过的指令的指令信息。

[0098] 可选地,在一种可能的实现方式中,协处理器每将一条指令信息中的未解码指令进行解码得到已解码的指令之后,还可以将该已解码的指令存储至指令缓存队列。相应的,协处理器可以从该指令缓存队列中读取第一指令。其中,该指令缓存队列用于存储已解码的多个指令。

[0099] 需要说明的是,该指令缓存队列可以为一个先进先出的队列,协处理器每解码得到一条指令,可以将该指令存入至该指令缓存队列的队尾。相应的,协处理器可以从该指令缓存队列的队首读取当前待分析的第一指令。通过该指令缓存队列存储多个已解码的指令,可以使得协处理器获取指令信息并解码指令的操作与读取已解码的指令进行污点分析的操作解耦,以流水线的方式并行执行,从而提高了协处理器进行污点分析的效率。

[0100] 步骤502:根据第一指令对应的污点传播规则,更新协处理器的多个寄存器中存储的第一指令的目的操作数所指示的第一通用寄存器或第一内存单元对应的污点信息,主处理器的多个通用寄存器包括第一通用寄存器,程序内存空间包括第一内存单元。

[0101] 在获取到第一指令之后,协处理器可以根据第一指令的操作码确定第一指令的指令类型,如果第一指令的指令类型不为污点检测类且不为污点标记类,则协处理器可以根据第一指令的指令类型从预先配置的多个污点传播规则中确定第一指令所对应的污点传播规则,并获取第一指令的源操作数对应的污点信息。之后,协处理器可以根据获取到的第一指令对应的污点传播规则和源操作数对应的污点信息来更新第一指令的目的操作数所

指示的第一通用寄存器或第一内存单元对应的污点信息,以此来完成第一指令的污点传播。

[0102] 需要说明的是,在本申请实施例中,根据指令的操作码所指示的指令所要执行的操作,可以将指令分成不同的类型。示例性的,指令类型可以包括五种,分别为算术逻辑单元(arithmetic and logic unit,ALU)类、加载(load)类、存储(store)类、污点检测类以及污点标记类。

[0103] 其中,ALU类指令是指用于实现算术逻辑运算的指令。例如,ALU类指令可以包括累加(add)指令、与(and)指令等。

[0104] load类指令是指用于实现向通用寄存器中加载数据的指令。例如,load类指令可以包括load指令、mov指令等。

[0105] store类指令是指用于实现向内存中存储数据的指令。例如,store类指令可以包括store指令、mov指令等。值得注意的是,根据mov指令中的源操作数和目的操作数的类型,mov指令可以用于将内存中的数据传送至寄存器,也可以用于将寄存器中的数据传送至内存,因此,mov指令的操作码对应的指令类型可以包括load类和store类。

[0106] 污点检测类指令可以是指触发污点检测的指令,例如,污点检测类指令可以包括用于实现跳转、输出的指令,如跳转(jump)指令。由于污点检测类指令是用于触发污点检测的,所以在本申请实施例中,污点检测类指令不参与污点传播,也即,对于污点检测类指令,协处理器可以直接基于该类指令进行污点检测,而不执行污点传播。

[0107] 污点标记类指令是用于为从外部引入的数据设置初始的污点信息的指令。由前述对污点信息的实现方式的介绍可知,在本申请实施例中,栈区中的内存单元对应的污点信息和通用寄存器对应的污点信息不仅包括用于指示存储的信息是否被污染的污点标签,还可以包括用于指示存储的信息为数据还是堆地址的信息类型标签。基于此,在本申请实施例中,污点标记类指令可以包括污点标记指令和地址标记指令。其中,污点标记指令用于为从外部引入的数据设置初始的污点标签,地址标记指令则用于标记通用寄存器或内存单元中存储的信息为堆地址,通过该污点标记指令和地址标记指令,可以实现前述介绍的污点信息的初始设置。

[0108] 例如,污点标记指令的指令格式可以为taint a,b。其中,a为目的操作数,b为源操作数,且a和b均为寄存器标识。基于此,当a为rs1,b为0时,则说明通用寄存器rs1中当前存储的信息被污染,在这种情况下,可以将影子寄存器文件中该rs1对应的污点信息中包括的污点标签设置为1,以指示rs1被污染。可选地,当a不为0,b为rs2时,则通用寄存器rs2中存储的是偏移量,其中,该偏移量用于指示程序内存空间中的内存单元。在这种情况下,基于rs2中的偏移量,可以将污点存储文件中存储的程序内存空间中该偏移量所指示的内存单元的污点信息包括的污点标签设置为1,以此来指示该内存单元中存储的信息被污染。

[0109] 可选地,上述a不为0时,例如,当a为rs1时,通用寄存器rs1中存储的信息可以为类型值,该类型值用于指示rs2中的偏移量所指示的内存单元为程序内存空间的栈区中的内存单元还是全局区中的内存单元。例如,当类型值为0,用于指示偏移量所指示的内存单元为栈区中的内存单元,当类型值为1,用于指示偏移量所指示的内存单元为全局区中的内存单元。这样,基于该类型值,协处理器可以确定是在污点存储文件中的第一子文件还是第二子文件中设置该内存单元的污点标签。

[0110] 地址标记指令的指令格式可以为src c,d。其中,c为目的操作数,d为源操作数,且c和d均为寄存器标识。当c为0,d为rs1时,则表示通用寄存器rs1中存放的是一个堆地址,在这种情况下,可以将影子寄存器文件中rs1对应的污点信息中包括的信息类型标签设置为1。可选地,当c不为0,d为rs2时,则通用寄存器rs2中存储的为偏移量,该偏移量用于指示程序内存空间中的内存单元。这样,基于rs2中的偏移量,可以将污点存储文件中存储的该偏移量所指示的内存单元的污点信息包括的信息类型标签设置为1,以此来指示该内存单元存储的为一个堆地址。

[0111] 可选地,上述c不为0时,例如,当c为rs1时,通用寄存器rs1中存储的信息可以为类型值,该类型值用于指示rs2中的偏移量所指示的内存单元为程序内存空间的栈区中的内存单元还是全局区中的内存单元。其中,该类型值的实现方式可以参考前述介绍的污点标记指令中的类型值的实现方式。这样,基于地址标记指令中的该类型值,协处理器可以确定是在第一子文件还是第二子文件中设置该内存单元对应的信息类型标签。

[0112] 值得注意的是,污点标记类指令由于是用于设置初始的污点信息的,因此,该污点标记类指令也不参与污点传播,也即,对于该污点标记类指令,协处理器不执行污点传播操作。另外,在本申请实施例中,主处理器在检测到污点标记类指令后,可以忽略该污点标记类指令,并生成污点标记类指令的执行日志,将该指令日志作为污点标记类指令的指令信息发送至协处理器。其中,污点标记类指令的执行日志可以包括未解码的指令、指令所使用的寄存器的地址、指令所使用的数据,其中,指令所使用的数据即包括指令所使用的寄存器中存储的信息,如偏移量。

[0113] 基于上述介绍,协处理器中可以预先配置有第一映射关系,该第一映射关系中可以包括上述的指令类型与对应的操作码。其中,一个操作码可能对应一个指令类型也可能对应多个指令类型,一个指令类型可以对应至少一个操作码。在此基础上,协处理器可以从该第一映射关系中获取第一指令的操作码对应的指令类型。如果获取到的指令类型为一个,则获取到的指令类型即为第一指令的指令类型。可选地,如果获取到的指令类型为多个,则协处理器可以根据第一指令的操作数从获取到的多个指令类型中选择一个作为该第一指令的指令类型。

[0114] 例如,对于mov指令,获取到的指令类型可以包括load类和store类,在这种情况下,协处理器可以根据mov指令的目的操作数来选择一个指令类型。例如,如果mov指令的目的操作数指示的为内存单元,则该第一指令的指令类型为store类。如果mov指令的目的操作数指示的为通用寄存器,则该第一指令的指令类型为load类。

[0115] 在获取到第一指令的指令类型之后,如果该第一指令的指令类型不为污点检测类且不为污点标记类,则协处理器可以基于该第一指令的指令类型获取第一指令对应的污点传播规则。

[0116] 示例性的,协处理器中可以预先配置有第二映射关系。该第二映射关系包括指令类型与对应的污点传播规则。基于此,协处理器可以从第二映射关系中获取第一指令的指令类型对应的污点传播规则,获取到的污点传播规则即为第一指令对应的污点传播规则。

[0117] 需要说明的是,由前述介绍可知,污点检测类指令和污点标记类不参与污点传播,因此,第二映射关系中可以包括ALU类、load类和store类对应的污点传播规则,而不包括污点检测类和污点标记类对应的污点传播规则。

[0118] 其中,对于ALU类指令,主处理器在执行该类指令时,所涉及的数据通常在通用寄存器之间传递,因此,ALU类对应的污点传播规则可以为源操作数所指示的通用寄存器对应的污点信息向目的操作数所指示的通用寄存器传播。

[0119] 对于load类指令,主处理器在执行该类指令时,数据将被传递至通用寄存器中,因此,load类对应的污点传播规则可以为源操作数对应的污点信息向目的操作数所指示的通用寄存器传播。

[0120] 对于store类指令,主处理器在执行该类指令时,数据将被传递至内存中,因此,store类对应的污点传播规则可以为源操作数对应的污点信息向目的操作数所指示的内存单元传播。

[0121] 协处理器除了获取第一指令对应的污点传播规则,还可以获取第一指令的源操作数对应的污点信息,之后,协处理器可以基于第一指令对应的污点传播规则以及第一指令的源操作数对应的污点信息,更新第一指令的目的操作数所指示的第一通用寄存器或第一内存单元对应的污点信息。

[0122] 由前述介绍可知,源操作数可以为一个立即数,也可以用于指示通用寄存器或内存单元。基于此,在第一种可能的情况中,如果第一指令的源操作数为一个立即数,则此时协处理器对应的多个寄存器中将不存在该源操作数对应的污点信息。并且,由于第一指令不为污点标记指令,所以协处理器可以确定该源操作数不为污点数据。在这种情况下,协处理器可以基于第一指令对应的污点传播规则,确定第一指令的目的操作数所指示的第一通用寄存器或第一内存单元中存储的信息也未被污染,进而将影子寄存器文件中第一通用寄存器对应的污点信息中的污点标签更新为0,或将污点存储文件中第一内存单元对应的污点信息中的污点标签更新为0。除此之外,由于源操作数为一个立即数,经过第一指令的指令码操作后,目的操作数所指示的第一通用寄存器或第一内存单元中存储的信息也将为数据,在这种情况下,协处理器还可以将第一通用寄存器或第一内存单元对应的污点信息中包括的信息类型标签更新为0,以此来指示第一通用寄存器或第一内存单元中存储的为数据。

[0123] 在第二种可能的情况中,如果第一指令的源操作数指示第二通用寄存器,则协处理器可以从影子寄存器文件中获取第二通用寄存器对应的污点信息。之后,协处理器可以基于第一指令对应的污点传播规则以及第二通用寄存器对应的污点信息,更新第一通用寄存器或第一内存单元对应的污点信息。

[0124] 示例性的,如果第一指令为ALU类指令,则第一指令的目的操作数可以用于指示第一通用寄存器。在这种情况下,协处理器可以将影子寄存器文件中第一通用寄存器对应的污点信息直接更新为第二通用寄存器对应的污点信息。

[0125] 可选地,如果第一指令为store类指令,则第一指令的目的操作数可以用于指示第一内存单元。在这种情况下,协处理器可以将污点存储文件中第一内存单元对应的污点信息中的污点标签更新为第二通用寄存器对应的污点信息中的污点标签,并将第一内存单元对应的污点信息中的信息类型标签更新为第二通用寄存器对应的污点信息中的信息类型标签。

[0126] 需要说明的是,第一指令的目的操作数可以指示第一内存单元的地址,因此,协处理器可以基于该目的操作数所指示的第一内存单元的地址,从污点存储文件中查找第一内

存单元对应的污点信息,进而对该第一内存单元对应的污点信息进行更新。

[0127] 可选地,当第二通用寄存器对应的污点信息中的信息类型标签指示第二通用寄存器中存储的是一个堆地址时,第二通用寄存器对应的污点信息中还将包括该堆地址在程序内存空间中的存储地址,该存储地址即为第一内存单元的地址。在这种情况下,协处理器也可以基于第二通用寄存器对应的污点信息中记录的该堆地址在程序内存空间中的存储地址,在污点存储文件中查找第一内存单元对应的污点信息,进而对该第一内存单元对应的污点信息进行更新。

[0128] 在第三种可能的情况中,如果第一指令的源操作数指示第二内存单元,则协处理器可以从污点存储文件中获取第二内存单元对应的污点信息。之后,协处理器可以基于第一指令对应的污点传播规则以及第二内存单元对应的污点信息,更新第一通用寄存器或第一内存单元对应的污点信息。

[0129] 其中,如果第一指令为load类指令,则第一指令的目的操作数可以指示第一通用寄存器。基于此,协处理器可以将影子寄存器文件中第一通用寄存器对应的污点信息中的污点标签更新为第二内存单元对应的污点信息中的污点标签。

[0130] 另外,如果第二内存单元对应的污点信息中的信息类型标签指示第二内存单元存储的信息为数据,则协处理器可以将第一通用寄存器对应的污点信息中的信息类型标签更新为第二内存单元对应的信息类型标签,并将第一通用寄存器对应的污点信息中除污点标签、信息类型标签之外的其他比特更新为0。如果第二内存单元对应的污点信息中的信息类型标签指示第二内存单元存储的信息为堆地址,则协处理器可以获取第二内存单元的地址,之后,将第一通用寄存器对应的污点信息中的信息类型标签更新为第二内存单元对应的信息类型标签,并将第一通用寄存器对应的污点信息中除污点标签、信息类型标签之外的其他比特更新为第二内存单元的地址。

[0131] 上述主要介绍了第一指令包括两个操作数,也即第一指令为双操作数指令的情况下,协处理器基于第一指令对应的污点传播规则进行污点传播的过程。在一些可能的情况中,第一指令也可能为单操作数指令,在这种情况下,第一指令包括的该操作数既为源操作数也为目的操作数,因此,协处理器同样可以基于上述的方法来进行污点传播。

[0132] 对于主处理器发送的任一个指令信息,协处理器在确定该指令信息对应的解码之后的指令不为污点检测类指令或污点标记类指令之后,均可以通过上述的方法对相应指令进行污点传播。而对于污点检测类指令,则协处理器可以通过步骤503进行污点检测。

[0133] 步骤503:根据多个寄存器中存储的污点信息,进行污点检测。

[0134] 示例性的,协处理器在通过上述方法对第一指令进行污点传播之后,可以通过前述步骤501介绍的方法获取第二指令。若第二指令为敏感指令且第二指令的操作数所指示的通用寄存器或内存单元对应的污点信息指示所存储的信息被污染,则协处理器可以向主处理器发送中断指令,并在指定内存区域内写入第二指令的程序计数值。

[0135] 其中,协处理器可以根据第二指令的操作码确定第二指令的指令类型。如果第二指令的指令类型为污点检测类,则可以确定第二指令为敏感指令,在这种情况下,协处理器可以从多个寄存器存储的污点信息中获取第二指令的操作数所指示的通用寄存器或内存单元对应的污点信息。其中,第二指令的操作数可以包括第二指令的源操作数和/或目的操作数。如果第二指令的操作数所指示的通用寄存器或内存单元对应的污点信息中的污点标

签指示相应的通用寄存器或内存单元被污染,则说明第二指令所执行的操作为异常操作,有可能产生信息安全问题。因此,协处理器可以向主处理器发送中断指令,以提示主处理器。另外,协处理器还可以将第二指令的指令信息中包括的程序计数值存储至指定内存区域中。其中,该指定内存区域为在内存中预先指定的用于存储最近一次检测到的执行异常操作的指令的程序计数值的区域。

[0136] 需要说明的是,由前述步骤501中的介绍可知,协处理器中可以包括一个指令信息队列,并且,在指令信息队列已满的情况下,协处理器也可以向主处理器发送中断指令。基于此,在本申请实施例中,主处理器在接收到中断指令之后,可以查看指定内存区域中是否存储有程序计数值,如果指定内存区域中存储有程序计数值,则可以确定是由执行异常操作的指令所触发的中断,在这种情况下,主处理器可以中断执行目标程序中的指令,并基于指定内存区域中的程序计数值进行后续的处理,在处理完成后删除指定内存区域中的程序计数值,并继续执行目标程序中的后续指令。可选地,如果指定内存区域中为空,则主处理器可以确定该中断指令是由于指令信息队列已满所触发的中断,在这种情况下,主处理器可以中断执行目标程序中的指令,并通过前述步骤501中介绍的方法查询指令信息队列是否已满,在查询到指令信息队列未满的情况下,继续执行目标程序中的后续指令。

[0137] 在本申请实施例中,计算机设备的协处理器包括多个寄存器,该多个寄存器不仅可以用于存储计算机设备的主处理器中的多个通用寄存器对应的污点信息,还可以用于存储主处理器当前执行的目标程序的程序内存空间对应的污点信息。这样,协处理器在对第一指令进行污点传播时,可以通过访问该多个寄存器来访问第一指令的操作数的污点信息,相较于通过访问内存来访问污点信息,操作更为简单,速度更快,性能开销更小。

[0138] 另外,在本申请实施例中,程序内存空间的栈区中的每个内存单元对应的污点信息不仅包括用于指示相应内存单元存储的信息是否被污染的污点标签,还包括用于指示相应内存单元存储的信息为数据或堆地址的信息类型标签。由于栈区中的堆地址可以为对象类型的变量在堆区中的起始地址,所以通过栈区中每个内存单元对应的污点信息不仅可以标记栈区中存储的数据的污染情况,还可以间接的标记堆区中的对象类型的变量的污染情况。也即,本申请实施例通过栈区中的每个内存单元对应的污点信息可以实现针对栈区的基于内存单元这一粒度的污点标记,同时,还可以实现针对堆区的基于变量粒度的污点标记。例如,当一个内存单元的大小为一个字节时,则通过栈区中每个内存单元对应的污点信息,可以实现针对栈区的字节粒度的污点标记,以及针对堆区的变量粒度的污点标记。这样,相较于为程序内存空间的栈区和堆区中的每个内存单元设置污点标签,减少了污点标签的数据量,从而减少了用于存储污点标签的存储空间。

[0139] 基于上述实施例中介绍的污点分析方法,本申请实施例还提供了一种部署有RSIC-V架构的主处理器控制对应的协处理器进行污点分析的详细流程。如图6所示,主处理器的处理核可以通过RoCC接口向协处理器发送指令的执行日志。协处理器可以对接收到指令的执行日志进行过滤,并将过滤后的与污点分析有关的指令的指令信息存放至指令信息队列Q1中。相应的,协处理器可以从指令信息队列中获取指令信息,并将该指令信息中的未解码的指令进行解码,将解码得到的指令存放至指令缓存队列Q2中。协处理器可以从指令缓存队列Q2中获取当前待分析的指令。若该指令不为污点检测类指令且不为污点标记类指令,则从预先设置的污点传播规则中获取该指令对应的污点传播规则,并基于该指令的污

点传播规则更新影子寄存器文件和污点存储文件,以完成污点传播。若该指令为污点检测类指令,则通过读取影子寄存器和/或污点存储文件中的污点信息进行污点检测,在基于读取的污点信息确定该指令所使用的通用寄存器或内存单元被污染的情况下,则可以认为检测到异常,在这种情况下,协处理器可以向主处理器的处理核发送中断指令,并将该指令的程序计数值写入至主处理器和协处理器的共享数据缓存中。若该指令为污点标记类指令,则协处理器可以基于该指令在影子寄存器文件或污点存储文件中写入该指令所使用的通用寄存器或内存单元对应的污点信息。其中,协处理器获取指令,基于指令进行污点标记、污点传播、污点检测的实现方式可以参考前述实施例中介绍的实现方式,本申请实施例在此不再赘述。

[0140] 需要说明的是,在本申请实施例中,上述的计算机设备中的协处理器可以是一个专门用于对主处理器中运行的程序进行动态污点分析的硬件加速器。基于此,为了使得协处理器能够正常工作,在使用该协处理器之前,首先可以对该协处理器进行相应的硬件部署和软件部署。

[0141] 示例性的,在进行硬件部署时,首先利用计算机设备上的开源操作系统生成源文件,该源文件用于对协处理器的FPGA进行编程,以使该协处理器的FPGA具有能够实现上述污点分析方法的逻辑电路。在生成源文件之后,可以利用集成设计组件加载并分析该源文件,将该源文件与集成设计组件提供的各种约束进行综合,最终生成比特流文件。将该比特流文件进行烧录,并将烧录上的文件加载到协处理器的FPGA中,以实现对协处理器的FPGA的编程,从而完成该协处理器的硬件部署。

[0142] 在进行软件部署时,首先获取包含有linux常用命令以及工具的源码包,并对该源码包进行编译。之后,设置初始RAM文件系统;编译linux内核;之后,编译引导加载(bootloader)程序;之后,修改交叉编译器,以使得修改后的交叉编译器能够在编译目标程序的过程中进行源代码插桩,得到插桩后的目标程序的可执行文件,供主处理器执行,除此之外,修改后的交叉编译器还可以将用于实现本申请实施例提供的污点分析方法的源代码编译为协处理器所能执行的可执行文件,以使协处理器可以通过执行该可执行文件来实现本申请实施例提供的污点分析方法。

[0143] 接下来对本申请实施例提供的污点分析装置进行介绍。

[0144] 图7是本申请实施例提供的一种污点分析装置700的结构示意图,该装置可以部署于前述实施例中的计算机设备的协处理器中,该协处理器包括多个寄存器,该多个寄存器用于存储计算机设备的主处理器中的多个通用寄存器对应的污点信息,以及主处理器当前执行的目标程序的程序内存空间对应的污点信息。如图7所示,该污点分析装置包括指令获取模块701和污点传播模块702。

[0145] 其中,指令获取模块701,用于执行前述实施例中的步骤501;

[0146] 污点传播模块702,用于执行前述实施例中的步骤502。

[0147] 可选地,程序内存空间包括栈区,程序内存空间对应的污点信息包括栈区中的内存单元对应的污点信息,栈区中的内存单元对应的污点信息包括污点标签和信息类型标签,内存单元对应的污点标签用于指示内存单元所存储的信息是否被污染,内存单元对应的信息类型标签用于指示内存单元所存储的信息为数据或堆地址。

[0148] 可选地,程序内存空间还包括全局区,程序内存空间对应的污点信息还包括全局

区中的内存单元对应的污点标签。

[0149] 可选地,通用寄存器对应的污点信息包括污点标签和信息类型标签,通用寄存器对应的污点标签用于指示通用寄存器所存储的信息是否被污染,通用寄存器对应的信息类型标签用于指示通用寄存器所存储的信息为数据或堆地址,当通用寄存器对应的信息类型标签指示通用寄存器所存储的信息为堆地址时,通用寄存器对应的污点信息还包括通用寄存器存储的堆地址在程序内存空间中的存储地址。

[0150] 可选地,指令获取模块701具体用于:

[0151] 从指令缓存队列中读取第一指令,指令缓存队列用于存储已解码的多个指令,多个指令为目标程序中主处理器已执行的指令。

[0152] 可选地,指令获取模块701还用于:

[0153] 从指令信息队列中读取第一指令信息,指令信息队列用于存储主处理器发送的多个指令信息,第一指令信息包括未解码指令;

[0154] 对第一指令信息包括的未解码指令进行解码,得到第一指令,并将第一指令存储至指令缓存队列。

[0155] 可选地,该装置700还用于:

[0156] 若指令信息队列已满,则向主处理器发送中断指令,中断指令用于指示主处理器中断发送指令信息。

[0157] 可选地,该装置700还包括:

[0158] 污点检测模块703,用于根据多个寄存器中存储的污点信息,进行污点检测。

[0159] 可选地,指令获取模块701还用于获取第二指令,第二指令为目标程序中的指令;污点检测模块703具体用于:若第二指令为敏感指令且第二指令的操作数所指示的通用寄存器或内存单元对应的污点信息指示所存储的信息被污染,则向主处理器发送中断指令,并在指定内存区域内写入第二指令的程序计数值。

[0160] 可选地,主处理器和协处理器为基于RISC-V架构的处理器。

[0161] 在本申请实施例中,计算机设备的协处理器包括多个寄存器,该多个寄存器不仅可以用于存储计算机设备的主处理器中的多个通用寄存器对应的污点信息,还可以用于存储主处理器当前执行的目标程序的程序内存空间对应的污点信息。这样,协处理器在对第一指令进行污点传播时,可以通过访问该多个寄存器来访问第一指令的操作数的污点信息,相较于通过访问内存来访问污点信息,操作更为简单,速度更快,性能开销更小。

[0162] 需要说明的是,上述实施例提供污点分析装置中对模块的划分是示意性的,仅仅为一种逻辑功能划分,实际实现时也可以有另外的划分方式,另外,在本申请各个实施例中的各功能模块可以集成在一个电路中,也可以是单独物理存在,也可以两个或两个以上模块集成为一个模块中。上述集成的模块既可以采用硬件的形式实现,也可以采用软件功能模块的形式实现。

[0163] 该集成的模块如果以软件功能模块的形式实现并作为独立的产品销售或使用时,可以存储在一个计算机可读取存储介质中。基于这样的理解,本申请实施例的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来,该计算机软件产品存储在一个存储介质中,包括若干指令用以使得一台计算机设备的处理器(processor)执行本申请各个实施例该方法的全部或部分步骤。而前述

的存储介质包括:U盘、移动硬盘、只读存储器(read-only memory,ROM)、随机存取存储器(random access memory,RAM)、磁碟或者光盘等各种可以存储程序代码的介质。

[0164] 另外,上述实施例提供的污点分析装置与污点分析方法实施例属于同一构思,其具体实现过程详见方法实施例,这里不再赘述。

[0165] 在上述实施例中,可以全部或部分地通过软件、硬件、固件或者其任意结合来实现。当使用软件实现时,可以全部或部分地以计算机程序产品的形式实现。所述计算机程序产品包括一个或多个计算机指令。在计算机上加载和执行所述计算机指令时,全部或部分地产生按照本申请实施例所述的流程或功能。所述计算机可以是通用计算机、专用计算机、计算机网络、或者其他可编程装置。所述计算机指令可以存储在计算机可读存储介质中,或者从一个计算机可读存储介质向另一个计算机可读存储介质传输,例如,所述计算机指令可以从一个网站站点、计算机、服务器或数据中心通过有线(例如:同轴电缆、光纤、数据用户线(digital subscriber line,DSL))或无线(例如:红外、无线、微波等)方式向另一个网站站点、计算机、服务器或数据中心进行传输。所述计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集成的服务器、数据中心等数据存储设备。所述可用介质可以是磁性介质(例如:软盘、硬盘、磁带)、光介质(例如:数字通用光盘(digital versatiledisc,DVD))、或者半导体介质(例如:固态硬盘(solid state disk,SSD))等。

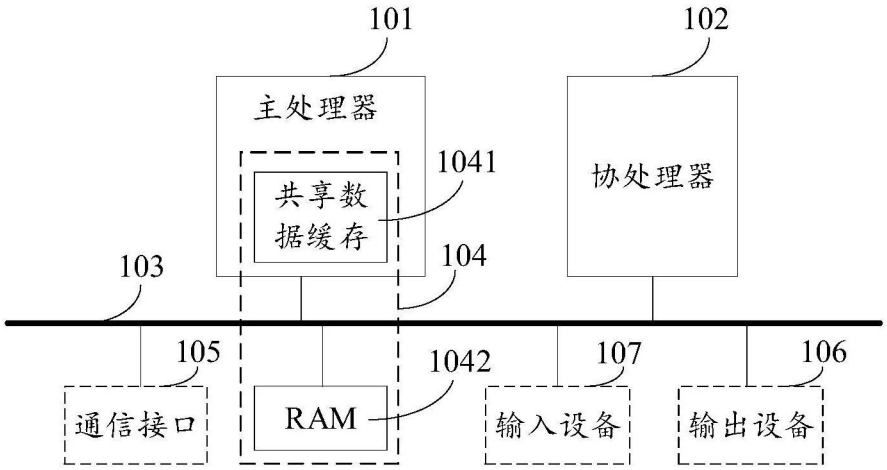

101
102
主处理器
1041
共享数据缓存
协处理器
103
104
105
1042
107
106
通信接口
RAM
输入设备
输出设备

图1

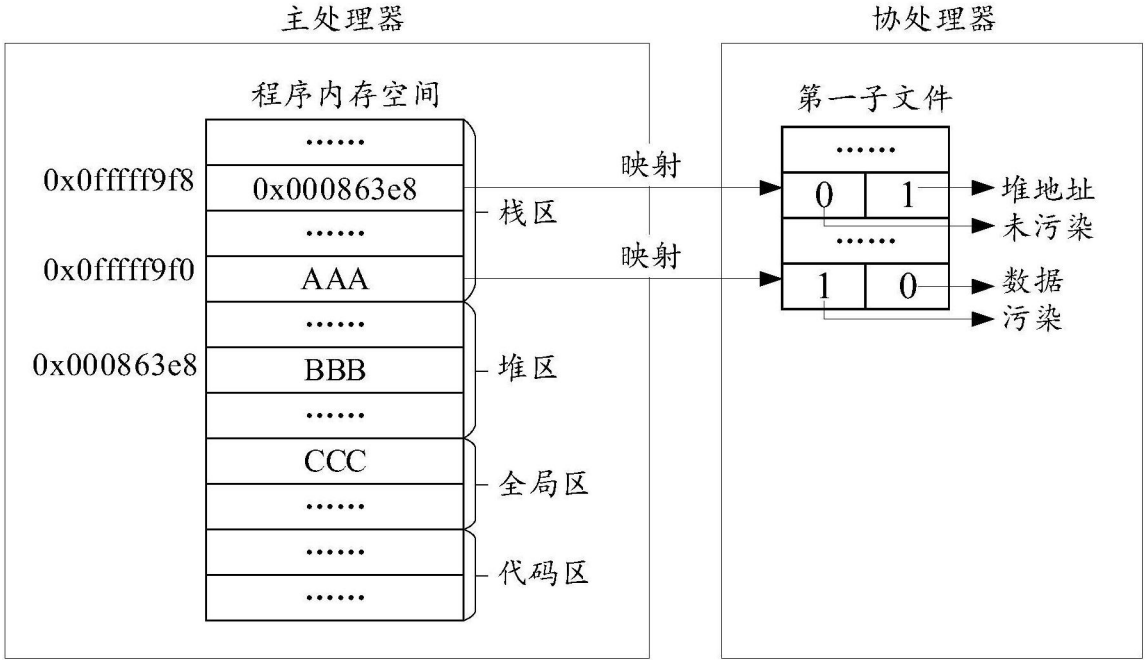

主处理器
协处理器
程序内存空间
第一子文件
0x0fffff9f8
0x000863e8
0x0fffff9f0
AAA
0x000863e8
BBB
CCC
栈区
堆区
全局区
代码区
映射
映射
堆地址
未污染
数据
污染

图2

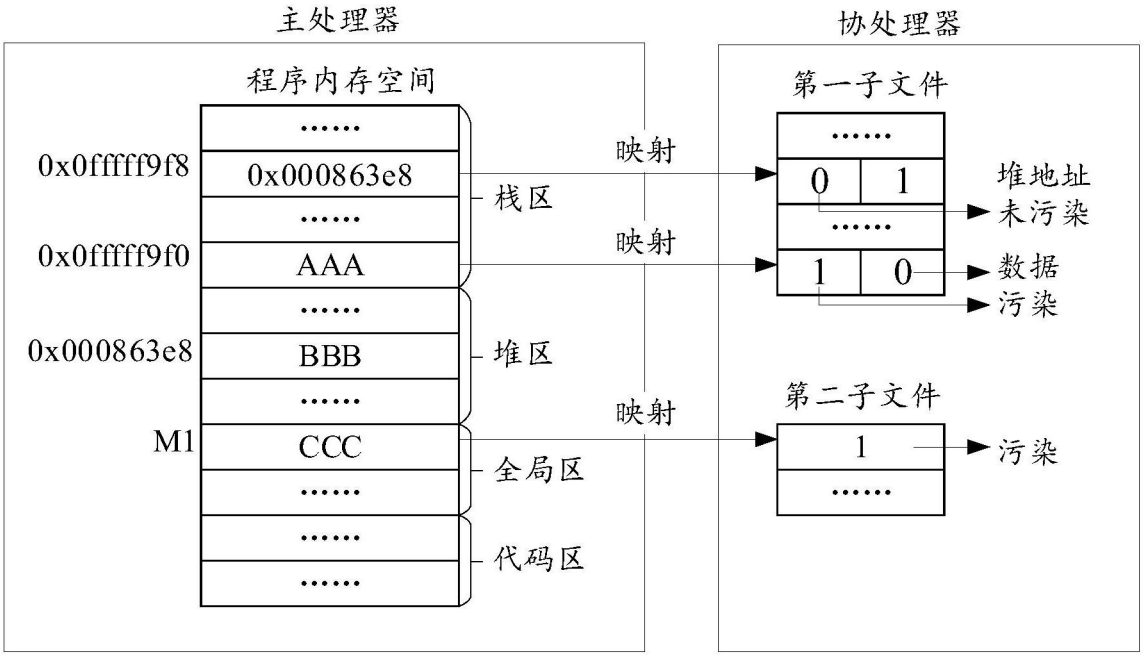

主处理器
协处理器
程序内存空间
第一子文件
0x0fffff9f8
0x000863e8
栈区
映射
0
1
堆地址
未污染
0x0fffff9f0
AAA
映射
1
0
数据
污染
0x000863e8
BBB
堆区
第二子文件
M1
CCC
映射
1
污染
全局区
代码区

图3

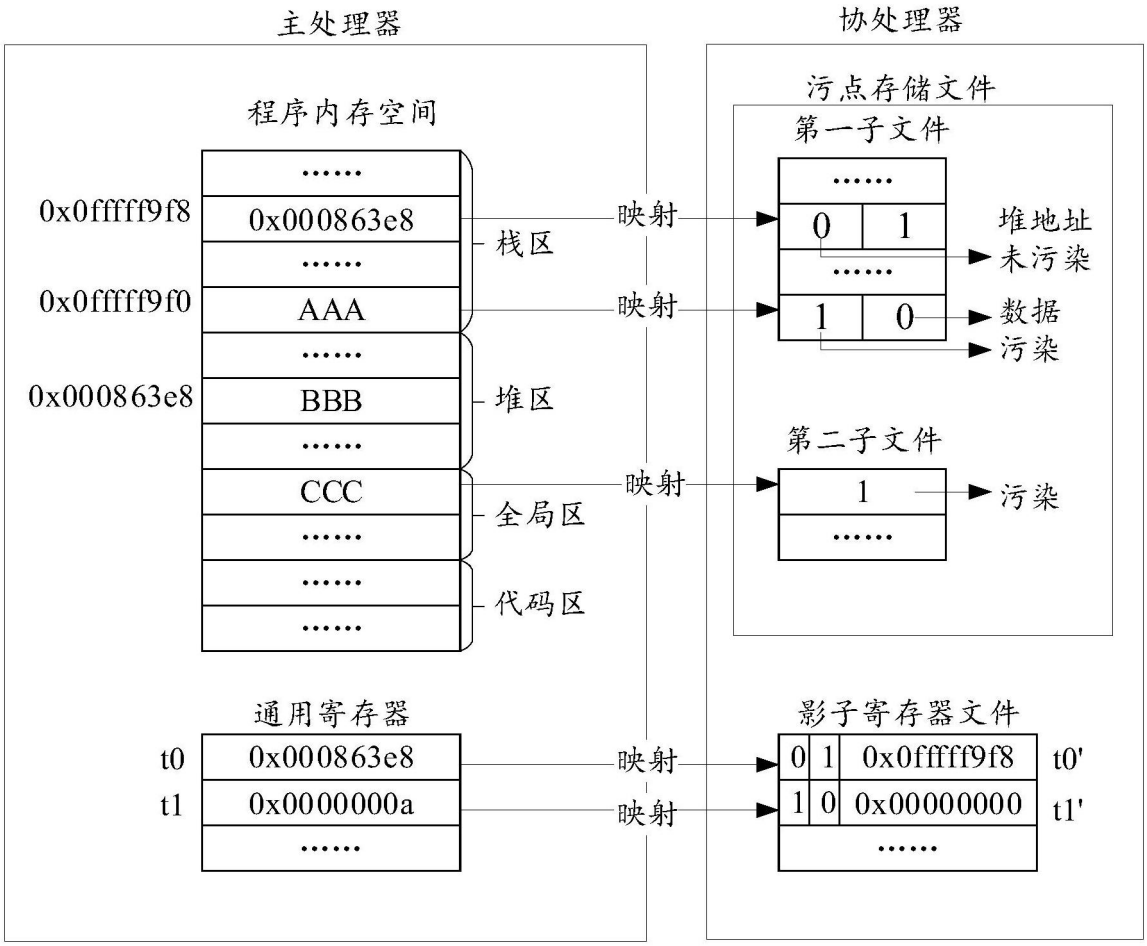

主处理器
协处理器
程序内存空间
污点存储文件
第一子文件
0x0fffff9f8
0x000863e8
栈区
0x0fffff9f0
AAA
堆区
0x000863e8
BBB
全局区
CCC
代码区
映射
堆地址
未污染
数据
污染
第二子文件
通用寄存器
影子寄存器文件
t0
t1
0x0000000a
0x0fffff9f8
0x00000000
t0'
t1'

图4

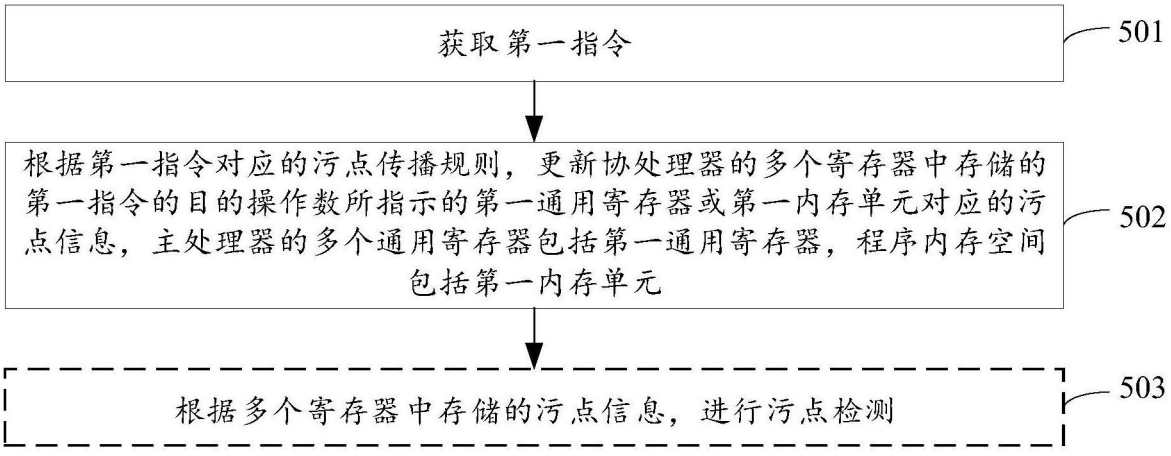

获取第一指令
501
根据第一指令对应的污点传播规则，更新协处理器的多个寄存器中存储的第一指令的目的操作数所指示的第一通用寄存器或第一内存单元对应的污点信息，主处理器的多个通用寄存器包括第一通用寄存器，程序内存空间包括第一内存单元
502
根据多个寄存器中存储的污点信息，进行污点检测
503

图5

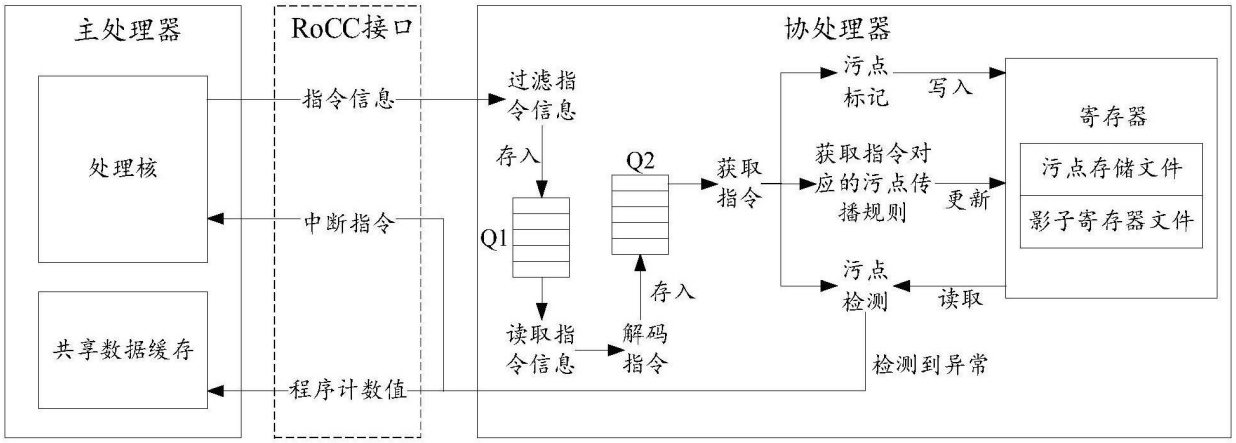

主处理器
处理核
共享数据缓存
RoCC接口
指令信息
中断指令
程序计数值
协处理器
过滤指令信息
存入
Q1
读取指令信息
Q2
存入
解码指令
获取指令
污点标记
写入
获取指令对应的污点传播规则
更新
污点检测
读取
检测到异常
寄存器
污点存储文件
影子寄存器文件

图6

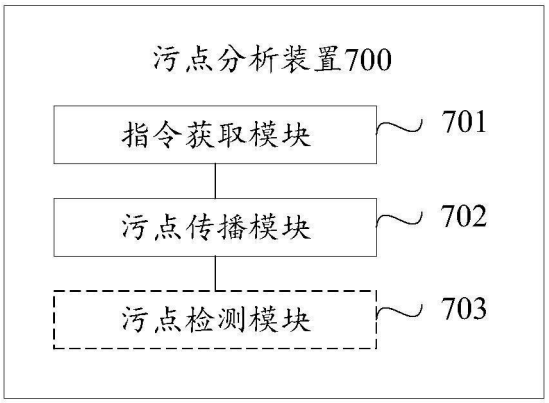

污点分析装置700
指令获取模块
701
污点传播模块
702
污点检测模块
703

图7