(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 120632857 A
(43)申请公布日 2025.09.12

(21)申请号 202510595407.1

(22)申请日 2025.05.09

(71)申请人 南方科技大学
地址 518055 广东省深圳市南山区桃源街道学苑大道1088号

(72)发明人 张锋巍 黄浩洋

(74)专利代理机构 广州嘉权专利商标事务所有限公司 44205
专利代理师 廖慧贤

(51)Int.Cl.
G06F 21/53(2013.01)
G06F 21/57(2013.01)

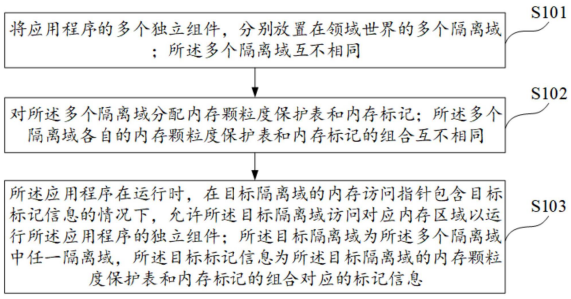
权利要求书2页 说明书15页 附图4页

(54)发明名称

可信执行环境内部隔离域设计方法、装置、设备以及介质

(57)摘要

本申请提供了一种可信执行环境内部隔离域设计方法、装置、设备以及介质，属于机密计算技术领域，该方法包括：将应用程序的多个独立组件，分别放置在领域世界的多个隔离域；多个隔离域互不相同；对所述多个隔离域分配内存颗粒度保护表和内存标记；多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同；应用程序在运行时，在目标隔离域的内存访问指针包含目标标记信息的情况下，允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件；目标隔离域为所述多个隔离域中任一隔离域，目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息。采用本申请技术方案能够实现组件级和线程级的隔离解决方案。



CN 120632857 A

1.一种可信执行环境内部隔离域设计方法,其特征在于,所述方法包括:

将应用程序的多个独立组件,分别放置在领域世界的多个隔离域;所述多个隔离域互不相同;

对所述多个隔离域分配内存颗粒度保护表和内存标记;所述多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同;

所述应用程序在运行时,在目标隔离域的内存访问指针包含目标标记信息的情况下,允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件;所述目标隔离域为所述多个隔离域中任一隔离域,所述目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息。

2.根据权利要求1所述的可信执行环境内部隔离域设计方法,其特征在于,所述对所述多个隔离域分配内存颗粒度保护表和内存标记,包括:

对所述多个隔离域分配内存颗粒度保护表;

在所述多个隔离域各自的内存颗粒度保护表相同的情况下,对所述多个隔离域分配不同的内存标记;

在所述多个隔离域各自的内存颗粒度保护表不相同的情况下,对所述多个隔离域分配相同或者不同的内存标记。

3.根据权利要求1所述的可信执行环境内部隔离域设计方法,其特征在于,对所述多个隔离域分配内存标记,包括:

获取待分配的初始内存标记;

在所述初始内存标记的数量小于所述多个隔离域的数量的情况下,基于所述内存颗粒度保护表对所述初始内存标记进行虚拟化得到目标内存标记,所述目标内存标记的数量大于或者等于所述多个隔离域的数量;

对所述多个隔离域分配所述目标内存标记。

4.根据权利要求1所述的可信执行环境内部隔离域设计方法,其特征在于,所述方法还包括:

对所述多个隔离域分配隔离域转换表;

基于所述隔离域转换表控制所述多个隔离域中的目标隔离域访问对应内存区域。

5.根据权利要求1所述的可信执行环境内部隔离域设计方法,其特征在于,所述方法还包括:

在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用。

6.根据权利要求5所述的可信执行环境内部隔离域设计方法,其特征在于,所述在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用,包括:

替换所述领域世界外部的不可信执行环境中的跳转指令,将所述不可信执行环境中的控制流转移至所述多个隔离域中的目标隔离域,以进行从所述不可信执行环境到所述多个隔离域的可信执行环境的第一函数调用。

7.根据权利要求5所述的可信执行环境内部隔离域设计方法,其特征在于,所述在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用,包

括：

在进行从所述可信执行环境到所述不可信执行环境的第二函数调用的情况下，将预设的附加指令添加到所述可信执行环境的跳转指令中，所述附加指令用于指示进行所述第二函数调用需要用于传递参数的寄存器的目标数量；

基于所述附加指令在通用寄存器中确定所述目标数量的目标寄存器，并将所述第二函数调用对应的参数保留在所述目标寄存器中，以及，对所述通用寄存器中除开所述目标寄存器以外其它寄存器上的数据进行擦除。

8.一种可信执行环境内部隔离域设计装置，其特征在于，所述装置包括：

程序划分模块，用于将应用程序的多个独立组件，分别放置在领域世界的多个隔离域；所述多个隔离域互不相同；

隔离模块，用于对所述多个隔离域分配内存颗粒度保护表和内存标记；所述多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同；

程序运行管理模块，用于所述应用程序在运行时，在目标隔离域的内存访问指针包含目标标记信息的情况下，允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件；所述目标隔离域为所述多个隔离域中任一隔离域，所述目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息。

9.一种计算机设备，其特征在于，所述计算机设备包括存储器和处理器，所述存储器存储有计算机程序，所述处理器执行所述计算机程序时实现权利要求1至7中任一项所述的可信执行环境内部隔离域设计方法。

10.一种计算机可读存储介质，所述计算机可读存储介质存储有计算机程序，其特征在于，所述计算机程序被处理器执行时实现权利要求1至7中任一项所述的可信执行环境内部隔离域设计方法。

可信执行环境内部隔离域设计方法、装置、设备以及介质

技术领域

[0001] 本申请涉及机密计算技术领域,尤其涉及一种可信执行环境内部隔离域设计方法、装置、设备以及介质。

背景技术

[0002] 随着移动计算和物联网技术的快速发展,电子产品处理器架构(Advanced RISC Machine,ARM)因其高能效比和灵活性,已成为智能手机、嵌入式设备以及云计算领域的主导架构。

[0003] 相关技术中,ARM的机密计算架构(Confidential Compute Architecture,CCA)是相比于TrustZone可信执行环境技术的新一代可信执行环境技术,CCA旨在提供更灵活、更安全的隔离解决方案,其在ARM架构设计层面引入了领域世界(Realm World),提供专门面向第三方开发者的虚拟机级可信执行环境。然而,尽管CCA相比TrustZone在易用性和灵活性方面取得了显著进展,但CCA采用的虚拟机级隔离方案也引入了新的安全风险:由于每个领域虚拟机需要运行完整的客户内核和多个应用程序组件,其潜在的攻击面显著扩大。

发明内容

[0004] 本申请实施例的主要目的在于提出一种可信执行环境内部隔离域设计方法、装置、设备以及介质,旨在实现组件级和线程级的隔离解决方案,从而降低领域虚拟机内组件中的漏洞针对整个系统的威胁。

[0005] 为实现上述目的,本申请实施例的第一方面提出了一种可信执行环境内部隔离域设计方法,所述方法包括:

[0006] 将应用程序的多个独立组件,分别放置在领域世界的多个隔离域;所述多个隔离域互不相同;

[0007] 对所述多个隔离域分配内存颗粒度保护表和内存标记;所述多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同;

[0008] 所述应用程序在运行时,在目标隔离域的内存访问指针包含目标标记信息的情况下,允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件;所述目标隔离域为所述多个隔离域中任一隔离域,所述目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息。

[0009] 在一些实施例中,所述对所述多个隔离域分配内存颗粒度保护表和内存标记,包括:

[0010] 对所述多个隔离域分配内存颗粒度保护表;

[0011] 在所述多个隔离域各自的内存颗粒度保护表相同的情况下,对所述多个隔离域分配不同的内存标记;

[0012] 在所述多个隔离域各自的内存颗粒度保护表不相同的情况下,对所述多个隔离域分配相同或者不同的内存标记。

[0013] 在一些实施例中,对所述多个隔离域分配内存标记,包括:

[0014] 获取待分配的初始内存标记;

[0015] 在所述初始内存标记的数量小于所述多个隔离域的数量的情况下,基于所述内存颗粒度保护表对所述初始内存标记进行虚拟化得到目标内存标记,所述目标内存标记的数量大于或者等于所述多个隔离域的数量;

[0016] 对所述多个隔离域分配所述目标内存标记。

[0017] 在一些实施例中,所述方法还包括:

[0018] 对所述多个隔离域分配隔离域转换表;

[0019] 基于所述隔离域转换表控制所述多个隔离域中的目标隔离域访问对应内存区域。

[0020] 在一些实施例中,所述方法还包括:

[0021] 在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用。

[0022] 在一些实施例中,所述在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用,包括:

[0023] 替换所述领域世界外部的不可信执行环境中的跳转指令,将所述不可信执行环境中的控制流转移至所述多个隔离域中的目标隔离域,以进行从所述不可信执行环境到所述多个隔离域的可信执行环境的第一函数调用。

[0024] 在一些实施例中,所述在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用,包括:

[0025] 在进行从所述可信执行环境到所述不可信执行环境的第二函数调用的情况下,将预设的附加指令添加到所述可信执行环境的跳转指令中,所述附加指令用于指示进行所述第二函数调用需要用于传递参数的寄存器的目标数量;

[0026] 基于所述附加指令在通用寄存器中确定所述目标数量的目标寄存器,并将所述第二函数调用对应的参数保留在所述目标寄存器中,以及,对所述通用寄存器中除开所述目标寄存器以外其它寄存器上的数据进行擦除。

[0027] 为实现上述目的,本申请实施例的第二方面提出了一种可信执行环境内部隔离域设计装置,所述装置包括:

[0028] 程序划分模块,用于将应用程序的多个独立组件,分别放置在领域世界的多个隔离域;所述多个隔离域互不相同;

[0029] 隔离模块,用于对所述多个隔离域分配内存颗粒度保护表和内存标记;所述多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同;

[0030] 程序运行管理模块,用于所述应用程序在运行时,在目标隔离域的内存访问指针包含目标标记信息的情况下,允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件;所述目标隔离域为所述多个隔离域中任一隔离域,所述目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息。

[0031] 为实现上述目的,本申请实施例的第三方面还提出了一种计算机设备,所述计算机设备包括存储器和处理器,所述存储器存储有计算机程序,所述处理器执行所述计算机程序时实现上述第一方面所述的可信执行环境内部隔离域设计方法。

[0032] 为实现上述目的,本申请实施例的第四方面提出了一种计算机可读存储介质,所

述计算机可读存储介质存储有计算机程序,所述计算机程序被处理器执行时实现上述第一方面所述的可信执行环境内部隔离域设计方法。

[0033] 为实现上述目的,本申请实施例的第五方面提出了一种计算机程序产品,所述计算机程序产品存储有计算机程序,所述计算机程序被处理器执行时实现上述第一方面所述的可信执行环境内部隔离域设计方法。

[0034] 本申请实施例提出的可信执行环境内部隔离域设计方法、可信执行环境内部隔离域设计装置、计算机设备、计算机可读存储介质以及计算机程序产品,其通过将应用程序的多个独立组件,分别放置在领域世界的多个隔离域;所述多个隔离域互不相同;对所述多个隔离域分配内存颗粒度保护表和内存标记;述多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同;所述应用程序在运行时,在目标隔离域的内存访问指针包含目标标记信息的情况下,允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件;所述目标隔离域为所述多个隔离域中任一隔离域,所述目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息。

[0035] 本申请实施例通过对应用程序进行拆分,然后将应用程序的多个独立组件,分别放置在领域世界不同的多个隔离域;并且,基于对多个隔离域分配内存颗粒度保护表和内存标记,且多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同;从而在应用程序在运行时,在多个隔离域中的任一目标隔离域的内存访问指针包含目标标记信息(目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息)的情况下,才允许该目标隔离域访问对应内存区域以运行应用程序的独立组件。如此,也就是说,本申请实施例能够支持细粒度的组件级和线程级隔离,从而可以基于对单个函数或者线程进行隔离,来避免攻击者通过寻找单个组件内的漏洞打破可信执行环境的隔离边界,即:本申请实施例能够实现组件级和线程级的隔离解决方案,从而降低领域虚拟机内组件中的漏洞针对整个系统的威胁。

附图说明

[0036] 图1为本申请实施例提供的可信执行环境内部隔离域设计方法在一些实施例当中的步骤流程示意图;

[0037] 图2为本申请实施例提供的可信执行环境内部隔离域设计方法在一些实施例中涉及的整体架构示意图;

[0038] 图3为为本申请实施例提供的可信执行环境内部隔离域设计方法在一些实施例中涉及的内存隔离机制的工作原理示意图;

[0039] 图4为本申请实施例提供的可信执行环境内部隔离域设计方法在另一些实施例当中的步骤流程示意图;

[0040] 图5为图1中步骤S102的细化步骤流程示意图;

[0041] 图6为本申请实施例提供的可信执行环境内部隔离域设计方法在又一些实施例当中的步骤流程示意图;

[0042] 图7为本申请实施例提供的可信执行环境内部隔离域设计装置的结构示意图;

[0043] 图8为本申请实施例提供的计算机设备的硬件结构示意图。

具体实施方式

[0044] 为了使本申请的目的、技术方案及优点更加清楚明白,以下结合附图及实施例,对本申请进行进一步详细说明。应当理解,此处所描述的具体实施例仅用以解释本申请,并不用于限定本申请。

[0045] 需要说明的是,虽然在装置示意图中进行了功能模块划分,在流程图中示出了逻辑顺序,但是在某些情况下,可以以不同于装置中的模块划分,或流程图中的顺序执行所示出或描述的步骤。说明书和权利要求书及上述附图中的术语“第一”、“第二”等是用于区别类似的对象,而不必用于描述特定的顺序或先后次序。

[0046] 除非另有定义,本文所使用的所有的技术和科学术语与属于本申请的技术领域的技术人员通常理解的含义相同。本文中所使用的术语只是为了描述本申请实施例的目的,不是旨在限制本申请。

[0047] 首先,对本申请实施例中涉及的专业技术术语进行简单说明。

[0048] 可信执行环境(Trusted Execution Environment,TEE)。

[0049] 可信执行环境TEE作为一种硬件辅助的安全技术,通过在硬件层面提供隔离机制,将安全敏感任务与不可信系统软件隔离,从而有效抵御来自操作系统、虚拟机监视器等特权软件的攻击。

[0050] TrustZone可信执行环境技术。

[0051] TrustZone是最早被广泛应用于采用ARM架构的电子设备(后文统一简称为ARM设备)的可信执行环境技术。TrustZone通过将系统划分为普通世界(Normal World)和安全世界(Secure World),利用硬件隔离机制确保普通世界无法直接访问安全世界的内存区域,从而为安全敏感任务提供了可靠的执行环境。

[0052] 机密计算架构(Confidential Compute Architecture,CCA)。

[0053] 机密计算架构CCA是在TrustZone技术的基础上推出的新一代可信执行环境技术,旨在提供更灵活、更安全的隔离解决方案。在架构设计层面,CCA引入了领域世界(Realm World),提供专门面向第三方开发者的虚拟机级可信执行环境。与TrustZone相比,领域世界支持开发者在无需修改应用源代码的情况下直接在领域虚拟机(Realm Virtual Machine,Realm VM)中运行复杂应用程序。这种设计使开发者能够充分利用现有的代码库和工具链,大幅提升开发效率并降低了迁移成本。在硬件层面,CCA引入了领域管理扩展(Realm Management Extension,RME)。

[0054] 领域管理扩展(Realm Management Extension,RME)。

[0055] 领域管理扩展RME是构建CCA硬件架构的基础。RME通过扩展内存管理单元(Memory Management Unit,MMU),引入了一种称为内存颗粒保护检查(Granule Protection Check,GPC)的机制,以防止来自处理器进行越界内存访问。在启用RME的系统中,物理内存被划分为多个内存颗粒(granules),每个颗粒代表一段连续的物理内存,通常大小为4KB。当MMU完成虚拟地址到物理地址的翻译后,GPC会通过遍历内存颗粒度保护表(Granule Protection Table,GPT)进行检查,以验证处理器是否有权限访问目标物理地址。

[0056] 内存颗粒度保护表(Granule Protection Table,GPT)。

[0057] 内存颗粒度保护表GPT是位于根世界的类似于页表的数据结构,仅有运行在最高特权等级的安全监视器能够对其进行修改。

[0058] 接下来,对本申请实施例的整体构思进行说明。

[0059] 随着移动计算和物联网技术的快速发展,电子产品处理器架构ARM因其高能效比和灵活性,已成为智能手机、嵌入式设备以及云计算领域的主导架构。然而,随着电子设备的普及和应用场景的复杂化,传统的安全机制已无法有效抵御日益复杂的攻击手段。这其中,传统的安全机制如地址空间布局随机化(ASLR)和控制流完整性保护(CFI)等等。

[0060] 相关技术中,尽管CCA相比TrustZone在易用性和灵活性方面取得了显著进展,但CCA采用的虚拟机级隔离方案也引入了新的安全风险。这是因为:由于每个领域虚拟机需要运行完整的客户内核和多个应用程序组件,其潜在的攻击面显著扩大,攻击者只需找到虚拟机内任一组件中的漏洞,就可以此作为跳板,最终突破TEE环境的安全边界。另外,相关技术还在CCA的基础上进一步提出了面向用户态应用的TEE方案——Shelter。Shelter为每个应用程序分配独立的内存空间,并利用硬件辅助的访问控制确保内存区域的独占性,从而构建出独立的用户态可信执行环境。

[0061] 然而,Shelter这类应用级隔离方案仍然面临着一定的安全挑战。这种将完整应用程序置于独立执行环境中的做法,虽然能够防御来自外部的特权攻击,但无法解决来自应用程序内部的安全威胁。攻击者可以通过利用应用程序内部组件或线程中的漏洞(如CVE-2016-2176,CVE-2016-8706以及CVE-2023-46852),绕过可信执行环境构建的隔离边界,直接对敏感代码与数据进行访问。

[0062] 为此,本申请实施例旨在设计一种高可扩展、细粒度的可信执行环境内部隔离域方案,提供组件级和线程级的隔离机制,以降低单个漏洞对整个系统的威胁。基于此,本申请实施例提供了一种可信执行环境内部隔离域设计方法、可信执行环境内部隔离域设计装置、计算机设备、计算机可读存储介质以及计算机程序产品,通过对应用程序进行拆分,然后将应用程序的多个独立组件,分别放置在领域世界不同的多个隔离域;并且,基于对多个隔离域分配内存颗粒度保护表和内存标记,且多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同;从而在应用程序在运行时,在多个隔离域中的任一目标隔离域的内存访问指针包含目标标记信息(目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息)的情况下,才允许该目标隔离域访问对应内存区域以运行应用程序的独立组件。如此,也就是说,本申请实施例能够支持细粒度的组件级和线程级隔离,从而可以基于对单个函数或者线程进行隔离,来避免攻击者通过寻找单个组件内的漏洞打破可信执行环境的隔离边界,即:本申请实施例能够实现组件级和线程级的隔离解决方案,从而降低领域虚拟机内组件中的漏洞针对整个系统的威胁。

[0063] 此外,由于应用程序的复杂性和高并发特性,应用程序可能集成了大量具有不同功能的软件组件,或是创建了数百个并发线程来处理不同用户的请求以提高吞吐量。为此,本申请实施例还能够为任意数量的软件组件或线程创建独立的隔离域,并保持较低的性能开销。

[0064] 接下来,基于上述本申请实施例的整体构思,提出本申请实施例提供的可信执行环境内部隔离域设计方法、可信执行环境内部隔离域设计装置、计算机设备、计算机可读存储介质以及计算机程序产品各自的具体实施例。首先,详细描述本申请实施例中的可信执行环境内部隔离域设计方法的各个具体的实施例。

[0065] 本申请实施例可以基于人工智能技术对相关的数据进行获取和处理。其中,人工

智能(Artificial Intelligence,AI)是利用数字计算机或者数字计算机控制的机器模拟、延伸和扩展人的智能,感知环境、获取知识并使用知识获得最佳结果的理论、方法、技术及应用系统。

[0066] 人工智能基础技术一般包括如传感器、专用人工智能芯片、云计算、分布式存储、大数据处理技术、操作/交互系统、机电一体化等技术。人工智能软件技术主要包括计算机视觉技术、机器人技术、生物识别技术、语音处理技术、自然语言处理技术以及机器学习/深度学习等几大方向。

[0067] 需要说明的是,在本申请的各个具体实施方式中,当涉及到需要根据用户信息、用户行为数据,用户历史数据以及用户位置信息等与用户身份或特性相关的数据进行相关处理时,都会先获得用户的许可或者同意,而且,对这些数据的收集、使用和处理等,都会遵守相关法律法规和标准。此外,当本申请实施例需要获取用户的敏感个人信息时,会通过弹窗或者跳转到确认页面等方式获得用户的单独许可或者单独同意,在明确获得用户的单独许可或者单独同意之后,再获取用于使本申请实施例能够正常运行的必要的用户相关数据。

[0068] 此外,本申请实施例提供的可信执行环境内部隔离域设计方法可应用于终端中,也可应用于服务器端中,还可以是运行于终端或服务器端中的软件。在一些实施例中,终端可以是采用ARM设备,例如采用ARM架构的智能手机、平板电脑、笔记本电脑、台式计算机等电子设备或者计算机设备;服务器端可以配置成独立的物理服务器,也可以配置成多个物理服务器构成的服务器集群或者分布式系统,还可以配置成提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、CDN以及大数据和人工智能平台等基础云计算服务的云服务器;软件可以是实现可信执行环境内部隔离域设计方法的应用等,但并不局限于以上形式。

[0069] 或者,本申请实施例还可以用于众多通用或专用的计算机系统环境或配置中。例如:智能手机、个人计算机、服务器计算机、手持设备或便携式设备、平板型设备、多处理器系统、基于微处理器的系统、置顶盒、可编程的消费计算机设备、网络PC、小型计算机、大型计算机、包括以上任何系统或设备的分布式计算环境等等。本申请实施例可以在由计算机执行的计算机可执行指令的一般上下文中描述,例如程序模块。一般地,程序模块包括执行特定任务或实现特定抽象数据类型的例程、程序、对象、组件、数据结构等等。也可以在分布式计算环境中实践本申请,在这些分布式计算环境中,由通过通信网络而被连接的远程处理设备来执行任务。在分布式计算环境中,程序模块可以位于包括存储设备在内的本地和远程计算机存储介质中。

[0070] 为便于理解和阐述,后文中均以ARM设备应用本申请实施例提供的可信执行环境内部隔离域设计方法为例进行详细的说明。上述任一形式主题应用本申请实施例提供的可信执行环境内部隔离域设计方法的实现,都可以参照后文阐述的ARM设备应用可信执行环境内部隔离域设计方法的过程。

[0071] 请参照图1,图1为本申请实施例提供的可信执行环境内部隔离域设计方法在一些实施例当中的步骤流程示意图。应当理解的是,虽然图1中示出了一些方法步骤的执行顺序,但基于实际应用的不同设计需要,本申请实施例提供的可信执行环境内部隔离域设计方法当然可以采用不同于图中所示出的方法步骤的执行顺序。即,图1所示方法步骤的顺序并不构成对本申请实施例提供的可信执行环境内部隔离域设计方法的执行逻辑顺序的限

定,其它任何基于图1所示方法步骤顺序的合理变化均应包含在本申请实施例提供的可信执行环境内部隔离域设计方法的保护范围之内。

[0072] 如图1所示,在一些实施例中,本申请实施例提供的可信执行环境内部隔离域设计方法,可以包括但不限于步骤S101至步骤S103。

[0073] 步骤S101:将应用程序的多个独立组件,分别放置在领域世界的多个隔离域;所述多个隔离域互不相同。

[0074] 需要说明的是,ARM设备可以采用机密计算架构CCA引入领域世界,该领域世界的多个隔离域互不相同。

[0075] ARM设备通过将应用程序划分为多个独立组件,然后将该多个组件分别放置在领域世界不同的多个隔离域当中。

[0076] 在一些实施例中,本申请实施例提供的可信执行环境内部隔离域设计方法涉及的整体架构可以如图2所示:应用程序可以被划分为多个独立组件,该多个独立组件放置在领域世界不同的多个隔离域中。多个隔离域中的每个隔离域都在独立的执行环境中运行,除了安全监视器外不信任任何外部组件,包括虚拟机管理器和操作系统在内的系统软件都被视为不可信并严格隔离。

[0077] 步骤S102:对所述多个隔离域分配内存颗粒度保护表和内存标记;所述多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同。

[0078] ARM设备可以在将应用程序的多个独立组件放置在多个隔离域之前,针对该多个隔离域分配内存颗粒度保护表和内存标记。或者,ARM设备也可以在将应用程序的多个独立组件放置在多个隔离域的同时,同步对该多个隔离域分配内存颗粒度保护表和内存标记。

[0079] 在一些实施例中,为精简安全监视器的代码量,ARM设备可以采用CCA的系统软件栈解耦设计思路,仅在安全监视器中引入安全敏感功能,而将进程调度、物理内存管理等非敏感服务委托给普通世界系统软件处理。基于此,ARM设备可以在安全监视器中扩展出额外的功能模块用于进行内存隔离、内存管理以及隔离域管理。基于此,ARM设备可以通过在安全监视器中扩展出额外的内存隔离模块进行内存颗粒度保护表GPT和内存标记的配置(对多个隔离域分配内存颗粒度保护表和内存标记),以此确保分配给每个隔离域的内存区域只能由相应的隔离域访问,防止不可信系统软件的未授权访问。

[0080] 在一些实施例中,ARM设备在通过内存隔离模块对多个隔离域分配内存颗粒度保护表和内存标记时,可以对第一隔离域分配第一内存颗粒度保护表和第一内存标记,以及,对第二隔离域分配第二内存颗粒度保护表和第二内存标记。这其中,第一隔离域和第二隔离域为多个隔离域中互不相同的隔离域,而第一内存颗粒度保护表和第一内存标记的组合,与第二内存颗粒度保护表和第二内存标记的组合也不相同。

[0081] 示例性地,ARM设备在通过内存隔离模块对多个隔离域分配内存颗粒度保护表和内存标记时,可以基于领域管理扩展和内存标记扩展建立一个两层内存访问控制机制,为多个隔离域中的每个隔离域分配唯一的内存颗粒度保护表GPT与内存标记组合。如此,在运行时,内存标记扩展会检查用于内存访问的指针是否在其高位包含相应的标记,如果指针中缺少相应的标记,内存标记扩展将阻止此类未授权的访问尝试,而如果指针中存在相应标记,内存标记扩展就会允许访问。

[0082] 需要说明的是,内存标记扩展提供了内存标记功能。

[0083] 在一些实施例中,内存标记扩展允许开发者将内存标记分配给16字节对齐的内存区域。

[0084] 步骤S103:所述应用程序在运行时,在目标隔离域的内存访问指针包含目标标记信息的情况下,允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件;所述目标隔离域为所述多个隔离域中任一隔离域,所述目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息。

[0085] ARM设备在将应用程序的多个独立组件分别放置在领域世界不同的多个隔离域,以及对多个隔离域分配内存颗粒度保护表和内存标记之后,由于多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同,从而在应用程序运行时,在多个隔离域中的任一目标隔离域在访问对应内存区域的情况下,如果该目标隔离域的内存访问指针包含目标标记信息(目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息),ARM设备才允许目标隔离域访问其对应的内存区域以运行应用程序的独立组件。反之,如果目标隔离域的内存访问指针不包含该目标标记信息,ARM设备便拦截该目标隔离域访问对应内存区域。

[0086] 示例性地,如图3所示的内存隔离机制的工作原理,以ARM设备配置了GPT1和地址标记1的隔离域1为例,通过以下三种场景说明双重防护机制(上述ARM设备基于领域管理扩展和内存标记扩展建立的两层内存访问控制机制)的有效性。场景1:当隔离域1访问内存区域1时,由于该内存区域1属于领域世界且内存标记为1,同时满足GPC的权限验证和内存标记检测器的标记匹配要求,因此内存访问成功完成。场景2:当攻击者试图从隔离域1访问GPT相同但内存标记不同的区域2时,内存标记检测器发现实际内存标记2与预期内存标记1不匹配,从而拦截该次内存访问。场景3:当攻击者尝试使用隔离域1中的代码访问内存标记相同但GPT不同的内存区域16时,由于该内存区域16在GPT1中被标记为不可访问,GPC将根据权限配置拦截该次内存访问。

[0087] 本申请实施例中,通过ARM设备将应用程序划分为多个独立组件,然后将该多个组件分别放置在领域世界不同的多个隔离域当中。以及,通过ARM设备针对多个隔离域分配内存颗粒度保护表和内存标记。从而在应用程序运行时,在多个隔离域中的任一目标隔离域在访问对应内存区域的情况下,如果该目标隔离域的内存访问指针包含该目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息,ARM设备才允许目标隔离域访问其对应的内存区域以运行应用程序的独立组件。

[0088] 如此,本申请实施例就能够支持细粒度的组件级和线程级隔离,从而可以基于对单个函数或者线程进行隔离,来避免攻击者通过寻找单个组件内的漏洞打破可信执行环境的隔离边界,即:本申请实施例能够实现组件级和线程级的隔离解决方案,从而降低领域虚拟机内组件中的漏洞针对整个系统的威胁。

[0089] 此外,本申请实施例只需要基于ARM设备的标准硬件特性就可以实现,并不需要修改原有硬件指令集和架构,从而可适配于任何支持领域管理扩展与内存标记扩展的ARM设备中。

[0090] 请参照图4,图4为本申请实施例提供的可信执行环境内部隔离域设计方法在另一些实施例当中的步骤流程示意图。

[0091] 在一些实施例中,如图4所示,上述的步骤S102中“对所述多个隔离域分配内存标

记”的步骤,可以包括但不限于如下所示的步骤S401至步骤S403。

[0092] 步骤S401:获取待分配的初始内存标记。

[0093] ARM设备在通过内存隔离模块进行内存颗粒度保护表GPT和内存标记的配置时,获取内存标记扩展提供的内存标记,并将该内存标记作为初始内存标记。

[0094] 在一些实施例中,初始内存标记可以为内存标记扩展提供的16个不同的内存标记。

[0095] 步骤S402:在所述初始内存标记的数量小于所述多个隔离域的数量的情况下,基于所述内存颗粒度保护表对所述初始内存标记进行虚拟化得到目标内存标记,所述目标内存标记的数量大于或者等于所述多个隔离域的数量。

[0096] ARM设备在通过内存隔离模块进行内存颗粒度保护表GPT和内存标记的配置时,如果需要分配内存标记的隔离域的数量较大,比如待分配内存标记的多个隔离域的数量大于初始内存标记的数量,则在此情况下,ARM设备基于内存颗粒度保护表GPT对初始内存标记进行虚拟化,从而得到数量大于或者等于多个隔离域的数量的目标内存标记。

[0097] 步骤S403:对所述多个隔离域分配所述目标内存标记。

[0098] ARM设备在通过内存颗粒度保护表GPT对内存标记扩展提供的初始内存标记进行虚拟化得到目标内存标记之后,则通过内存隔离模块进一步对多个隔离域分配这些目标内存标记。

[0099] 在本实施例中,考虑到内存标记扩展仅提供16个不同的内存标记,从而无法支持开发者创建任意数量的隔离域,而为了解决这一问题,本实施例通过内存颗粒度保护GPT对内存标记扩展提供的内存标记进行虚拟化,如此就可以2.支持开发者创建任意数量的内存隔离域。

[0100] 请参照图5,图5为图1中步骤S102的细化步骤流程示意图。

[0101] 在一些实施例中,如图5所示,上述的步骤S102:对所述多个隔离域分配内存颗粒度保护表和内存标记,可以包括但不限于如下所示的步骤S501至步骤S503。

[0102] 步骤S501:对所述多个隔离域分配内存颗粒度保护表。

[0103] ARM设备在通过内存隔离模块进行内存颗粒度保护表GPT和内存标记的配置时,可以先通过内存隔离模块对多个隔离域分配内存颗粒度保护表GPT。

[0104] 在一些实施例中,ARM设备在通过内存隔离模块对多个隔离域分配内存颗粒度保护表GPT时,可以将不同的多个隔离域分配至同一个内存颗粒度保护表GPT,从而使得多个隔离域各自的内存颗粒度保护表相同,或者,也可以分别将不同的隔离域分配至不同的内存颗粒度保护表GPT,从而使得多个隔离域各自的内存颗粒度保护表不相同。

[0105] 步骤S502:在所述多个隔离域各自的内存颗粒度保护表相同的情况下,对所述多个隔离域分配不同的内存标记。

[0106] ARM设备在通过内存隔离模块将不同的多个隔离域分配至同一个内存颗粒度保护表GPT,从而使得多个隔离域各自的内存颗粒度保护表相同的情况下,ARM设备在通过内存隔离模块对多个隔离域分配内存标记时,就可以为多个隔离域分配不同的内存标记。

[0107] 步骤S503:在所述多个隔离域各自的内存颗粒度保护表不相同的情况下,对所述多个隔离域分配相同或者不同的内存标记。

[0108] ARM设备在通过内存隔离模块分别将不同的多个隔离域分配至不同的内存颗粒度

保护表GPT,从而使得多个隔离域各自的内存颗粒度保护表不相同的情况下,ARM设备在通过内存隔离模块对多个隔离域分配内存标记时,既可以为多个隔离域分配相同的内存标记,也可以为多个隔离域分配不同的内存标记。

[0109] 示例性地,ARM设备在通过内存隔离模块进行内存颗粒度保护表GPT和内存标记的配置时,对于位于同一内存颗粒度保护表GPT内的隔离域,可以通过分配不同的内存标记实现细粒度隔离。而对于位于不同内存颗粒度保护表GPT的隔离域,由于它们GPT对应的可访问物理内存互不重叠,因此可以安全地对内存标记进行复用,即,可以通过分配不同的内存标记实现细粒度隔离,也可以可以分配相同的内存标记也能够实现细粒度隔离。基于此,当恶意代码试图跨内存颗粒度保护表GPT访问同标记内存时,GPC会对其进行拦截。

[0110] 需要说明的是,由于内存颗粒度保护表GPT的创建没有数量限制,因此内存颗粒度保护表GPT与内存标记的正交组合可以支持开发者创建任意数量的隔离域。

[0111] 在本实施例中,通过将领域管理扩展与内存标记扩展相结合,并借助内存颗粒度保护表GPT对内存标记进行虚拟化,从而可以在保障细粒度内存访问控制的同时,支持创建任意数量的隔离域。

[0112] 请参照图6,图6为本申请实施例提供的可信执行环境内部隔离域设计方法在又一些实施例当中的步骤流程示意图。

[0113] 在一些实施例中,如图6所示,本申请实施例提供的可信执行环境内部隔离域设计方法还可以包括但不限于如下所示的步骤S601和步骤S602。

[0114] 步骤S601:对所述多个隔离域分配隔离域转换表;

[0115] 步骤S602:基于所述隔离域转换表控制所述多个隔离域中的目标隔离域访问对应内存区域。

[0116] ARM设备在安全监视器中扩展额外的功能模块用于内存管理时,可以通过在该安全监视器中扩展内存管理模块,引入隔离域转换表(Domain Translation Table,DTT)作为影子页表,通过对多个隔离域分配隔离域转换表DTT,并基于该隔离域转换表DTT控制多个隔离域中的目标隔离域访问对应内存区域,使得该多个隔离域的所有内存访问都受到影子页表的严格控制。

[0117] 在一些实施例中,ARM设备可以通过内存管理模块负责物理页的安全映射与状态跟踪,从而通过内存管理模块引入隔离域转换表DTT作为影子页表。

[0118] 需要说明的是,ARM设备的物理页的分配可以由普通世界的系统软件完成。考虑到如果将页表维护也交由普通世界的系统软件完成,那么攻击者可能通过篡改页表来发起攻击,对隔离域造成破坏,因此,为保护隔离域内存映射的完整性,ARM设备便引入了隔离域转换表DTT作为影子页表。

[0119] 在一些实施例中,每个DTT在对应隔离域的内存颗粒度保护表GPT中被标记为领域世界可访问,并在其他内存颗粒度保护表GPT中被标记为不可访问。此外,DTT仅允许安全监视器进行修改,从而确保其不被攻击者篡改。

[0120] 在一些实施例中,ARM设备可以通过动态切换页表来实现隔离域的安全执行。例如,在进入隔离域(切换至隔离域的可信执行环境)之前,安全监视器将页表基址寄存器配置为目标隔离域的DTT,使得该隔离域的所有内存访问都受到影子页表的严格控制。当隔离域退出时,安全监视器恢复原始页表配置,以保证不可信系统软件的正常运行。

[0121] 在一些实施例中,本申请实施例提供的可信执行环境内部隔离域设计方法,还可以包括如下所示的步骤:

[0122] 在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用。

[0123] ARM设备在安全监视器中扩展额外的功能模块用于隔离域管理时,可以通过在该安全监视器中扩展隔离域管理模块,从而基于该隔离域管理模块在多个隔离域的可信执行环境与领域世界外部的不可信执行环境之间进行函数调用。

[0124] 需要说明的是,隔离域管理模块可以作为跨域交互枢纽,负责在隔离域切换期间保存和恢复隔离域的上下文,并处理隔离域与不可信软件组件之间的调用转发。

[0125] 在一些实施例中,为了与不可信系统软件兼容,ARM设备可以通过隔离域管理模块提供一组隔离域管理接口(Domain ManagementInterface,DMI),允许不可信系统软件管理隔离域的生命周期和系统资源。

[0126] 在一些实施例中,上述的“在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用”的步骤,可以包括但不限于如下所示的步骤:

[0127] 替换所述领域世界外部的不可信执行环境中的跳转指令,将所述不可信执行环境中的控制流转移至所述多个隔离域中的目标隔离域,以进行从所述不可信执行环境到所述多个隔离域的可信执行环境的第一函数调用。

[0128] ARM设备在通过隔离域管理模块进行可信执行环境与不可信执行环境之间进行函数调用时,对于不可信执行环境到隔离域可信执行环境的调用(第一函数调用),可以通过对不可信执行环境中的跳转指令进行替换,从而将不可信执行环境中的控制流转移至隔离域中,例如将控制流转移至多个隔离域中的某一个或者多个目标隔离域。

[0129] 在一些实施例中,对于隔离域的可信执行环境到不可信执行环境的调用(第二函数调用),ARM设备可以采用陷入-转发的方法来实现自动转发。

[0130] 基于此,上述的“在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用”,可以包括但不限于如下所示的步骤:

[0131] 在进行从所述可信执行环境到所述不可信执行环境的第二函数调用的情况下,将预设的附加指令添加到所述可信执行环境的跳转指令中,所述附加指令用于指示进行所述第二函数调用需要用于传递参数的寄存器的目标数量;

[0132] 基于所述附加指令在通用寄存器中确定所述目标数量的目标寄存器,并将所述第二函数调用对应的参数保留在所述目标寄存器中,以及,对所述通用寄存器中除开所述目标寄存器以外其它寄存器上的数据进行擦除。

[0133] 需要说明的是,由于不可信代码并未在隔离域的影子页表中建立内存映射,任何从隔离域到不可信代码的函数调用跳转都会触发指令中止异常。安全监视器捕获此异常后,验证故障地址是否在隔离域可信代码的起始地址范围之外。如果是,安全监视器将调用转发至不可信环境中的对应函数。为了防止敏感数据泄漏,ARM设备还在通过隔离域管理模块进行可信执行环境到不可信执行环境的第二函数调用时,确保用于传递参数的通用寄存器不会在无意中暴露敏感信息。即,对于需要跳转到不可信代码的函数调用(第二函数调用),ARM设备会在可信执行环境的跳转指令前插入一条附加指令(例如mov x14,#arg_num)用于存储用于传递参数的寄存器数量(目标数量)。当安全监视器捕获从隔离域发起的第二

函数调用时,会基于附加指令检查有多少寄存器需要用于传递参数,然后,安全监视器仅将预期参数(第二函数调用对应的参数)保留在通用寄存器(目标寄存器)中,并对其他的通用寄存器(其它寄存器)中的数据进行擦除,以确保寄存器中存储的敏感数据不会被泄露。

[0134] 在本实施例中,通过ARM设备在安全监视器中扩展隔离域管理模块,以基于该隔离域管理模块在多个隔离域的可信执行环境与领域世界外部的不可信执行环境之间进行函数调用,从而实现了双向函数调用转发机制,使得隔离域与不可信环境之间能够透明地互相调用函数,进而可以进一步降低现有应用的适配成本,增强系统兼容性。

[0135] 接下来,请参阅图7,本申请实施例还提供一种可信执行环境内部隔离域设计装置,可信执行环境内部隔离域设计装置可以实现上述的可信执行环境内部隔离域设计方法。

[0136] 在一些实施例中,如图7所示,本申请实施例提供的可信执行环境内部隔离域设计装置,可以包括:

[0137] 程序划分模块,用于将应用程序的多个独立组件,分别放置在领域世界的多个隔离域;所述多个隔离域互不相同;

[0138] 隔离模块,用于对所述多个隔离域分配内存颗粒度保护表和内存标记;所述多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同;

[0139] 程序运行管理模块,用于所述应用程序在运行时,在目标隔离域的内存访问指针包含目标标记信息的情况下,允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件;所述目标隔离域为所述多个隔离域中任一隔离域,所述目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息。

[0140] 在一些实施例中,所述的隔离模块,还用于对所述多个隔离域分配内存颗粒度保护表;在所述多个隔离域各自的内存颗粒度保护表相同的情况下,对所述多个隔离域分配不同的内存标记;以及,在所述多个隔离域各自的内存颗粒度保护表不相同的情况下,对所述多个隔离域分配相同或者不同的内存标记。

[0141] 在一些实施例中,所述的隔离模块,还用于获取待分配的初始内存标记;在所述初始内存标记的数量小于所述多个隔离域的数量的情况下,基于所述内存颗粒度保护表对所述初始内存标记进行虚拟化得到目标内存标记,所述目标内存标记的数量大于或者等于所述多个隔离域的数量;以及,对所述多个隔离域分配所述目标内存标记。

[0142] 在一些实施例中,本申请实施例提供的可信执行环境内部隔离域设计装置,还可以包括:

[0143] 管理模块,用于对所述多个隔离域分配隔离域转换表;以及,基于所述隔离域转换表控制所述多个隔离域中的目标隔离域访问对应内存区域。

[0144] 在一些实施例中,所述管理模块,还用于在所述多个隔离域的可信执行环境与所述领域世界外部的不可信执行环境之间进行函数调用。

[0145] 在一些实施例中,所述管理模块,还用于替换所述领域世界外部的不可信执行环境中的跳转指令,将所述不可信执行环境中的控制流转移至所述多个隔离域中的目标隔离域,以进行从所述不可信执行环境到所述多个隔离域的可信执行环境的第一函数调用。

[0146] 在一些实施例中,所述管理模块,还用于在进行从所述可信执行环境到所述不可信执行环境的第二函数调用的情况下,将预设的附加指令添加到所述可信执行环境的跳转

指令中,所述附加指令用于指示进行所述第二函数调用需要用于传递参数的寄存器的目标数量;以及,基于所述附加指令在通用寄存器中确定所述目标数量的目标寄存器,并将所述第二函数调用对应的参数保留在所述目标寄存器中,以及,对所述通用寄存器中除开所述目标寄存器以外其它寄存器上的数据进行擦除。

[0147] 需要说明的是,本申请实施例提供的可信执行环境内部隔离域设计装置的具体实施方式,与上述可信执行环境内部隔离域设计方法的具体实施例基本相同,在此不再赘述。

[0148] 本申请实施例还提供了一种计算机设备,计算机设备包括存储器和处理器,存储器存储有计算机程序,处理器执行计算机程序时实现上述可信执行环境内部隔离域设计方法和可信执行环境内部隔离域设计方法。该计算机设备可以为采用ARM架构的任意ARM设备,例如智能手机、平板电脑、个人计算机等电子设备。

[0149] 请参阅图8,图8示意了一些实施例中的计算机设备的硬件结构,计算机设备可以包括:

[0150] 处理器801,可以采用通用的CPU(CentralProcessingUnit,中央处理器)、微处理器、应用专用集成电路(ApplicationSpecificIntegratedCircuit,ASIC)、或者一个或多个集成电路等方式实现,用于执行相关程序,以实现本申请实施例所提供的技术方案;

[0151] 存储器802,可以采用只读存储器(ReadOnlyMemory,ROM)、静态存储设备、动态存储设备或者随机存取存储器(RandomAccessMemory,RAM)等形式实现。存储器802可以存储操作系统和其他应用程序,在通过软件或者固件来实现本说明书实施例所提供的技术方案时,相关的程序代码保存在存储器802中,并由处理器801来调用执行本申请实施例的可信执行环境内部隔离域设计方法;

[0152] 输入/输出接口803,用于实现信息输入及输出;

[0153] 通信接口804,用于实现本设备与其他设备的通信交互,可以通过有线方式(例如USB、网线等)实现通信,也可以通过无线方式(例如移动网络、WIFI、蓝牙等)实现通信;

[0154] 总线805,在设备的各个组件(例如处理器801、存储器802、输入/输出接口803和通信接口804)之间传输信息;

[0155] 其中处理器801、存储器802、输入/输出接口803和通信接口804通过总线805实现彼此之间在设备内部的通信连接。

[0156] 本申请实施例还提供了一种计算机可读存储介质,该计算机可读存储介质存储有计算机程序,该计算机程序被处理器执行时实现上述可信执行环境内部隔离域设计方法和可信执行环境内部隔离域设计方法。

[0157] 存储器作为一种非暂态计算机可读存储介质,可用于存储非暂态软件程序以及非暂态性计算机可执行程序。此外,存储器可以包括高速随机存取存储器,还可以包括非暂态存储器,例如至少一个磁盘存储器件、闪存器件、或其他非暂态固态存储器件。在一些实施方式中,存储器可选包括相对于处理器远程设置的存储器,这些远程存储器可以通过网络连接至该处理器。上述网络的实例包括但不限于互联网、企业内部网、局域网、移动通信网及其组合。

[0158] 本申请实施例还提供了一种计算机程序产品,该计算机程序产品存储有计算机程序,该计算机程序被处理器执行时实现上述可信执行环境内部隔离域设计方法和可信执行环境内部隔离域设计方法。

[0159] 本申请实施例描述的实施例是为了更加清楚的说明本申请实施例的技术方案,并不构成对于本申请实施例提供的技术方案的限定,本领域技术人员可知,随着技术的演变和新应用场景的出现,本申请实施例提供的技术方案对于类似的技术问题,同样适用。

[0160] 本领域技术人员可以理解的是,图中示出的技术方案并不构成对本申请实施例的限定,可以包括比图示更多或更少的步骤,或者组合某些步骤,或者不同的步骤。

[0161] 以上所描述的装置实施例仅仅是示意性的,其中作为分离部件说明的单元可以是或者也可以不是物理上分开的,即可以位于一个地方,或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部模块来实现本实施例方案的目的。

[0162] 本领域普通技术人员可以理解,上文中所公开方法中的全部或某些步骤、系统、设备中的功能模块/单元可以被实施为软件、固件、硬件及其适当的组合。

[0163] 本申请的说明书及上述附图中的术语“第一”、“第二”、“第三”、“第四”等(如果存在)是用于区别类似的对象,而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换,以便这里描述的本申请的实施例能够以除了在这里图示或描述的那些以外的顺序实施。此外,术语“包括”和“具有”以及他们的任何变形,意图在于覆盖不排他的包含,例如,包含了一系列步骤或单元的过程、方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元,而是可包括没有清楚地列出的或对于这些过程、方法、产品或设备固有的其它步骤或单元。

[0164] 应当理解,在本申请中,“至少一个(项)”是指一个或者多个,“多个”是指两个或两个以上。“和/或”,用于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:只存在A,只存在B以及同时存在A和B三种情况,其中A,B可以是单数或者复数。字符“/”一般表示前后关联对象是一种“或”的关系。“以下至少一项(个)”或其类似表达,是指这些项中的任意组合,包括单项(个)或复数项(个)的任意组合。例如,a,b或c中的至少一项(个),可以表示:a,b,c,“a和b”,“a和c”,“b和c”,或“a和b和c”,其中a,b,c可以是单个,也可以是多个。

[0165] 在本申请所提供的几个实施例中,应该理解到,所揭露的装置和方法,可以通过其它的方式实现。例如,以上所描述的装置实施例仅仅是示意性的,例如,上述单元的划分,仅仅为一种逻辑功能划分,实际实现时可以有另外的划分方式,例如多个单元或组件可以结合或者可以集成到另一个系统,或一些特征可以忽略,或不执行。另一点,所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口,装置或单元的间接耦合或通信连接,可以是电性,机械或其它的形式。

[0166] 上述作为分离部件说明的单元可以是或者也可以不是物理上分开的,作为单元显示的部件可以是或者也可以不是物理单元,即可以位于一个地方,或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部单元来实现本实施例方案的目的。

[0167] 另外,在本申请各个实施例中的各功能单元可以集成在一个处理单元中,也可以是各个单元单独物理存在,也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现,也可以采用软件功能单元的形式实现。

[0168] 集成的单元如果以软件功能单元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,包括多指令用以使得一台计算机设备(可以是个人计算机,服务器,或者网络设备等)执行本申请各个实施例的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,简称ROM)、随机存取存储器(Random Access Memory,简称RAM)、磁碟或者光盘等各种可以存储程序的介质。

[0169] 以上参照附图说明了本申请实施例的优选实施例,并非因此局限本申请实施例的权利范围。本领域技术人员不脱离本申请实施例的范围和实质内所作的任何修改、等同替换和改进,均应在本申请实施例的权利范围之内。

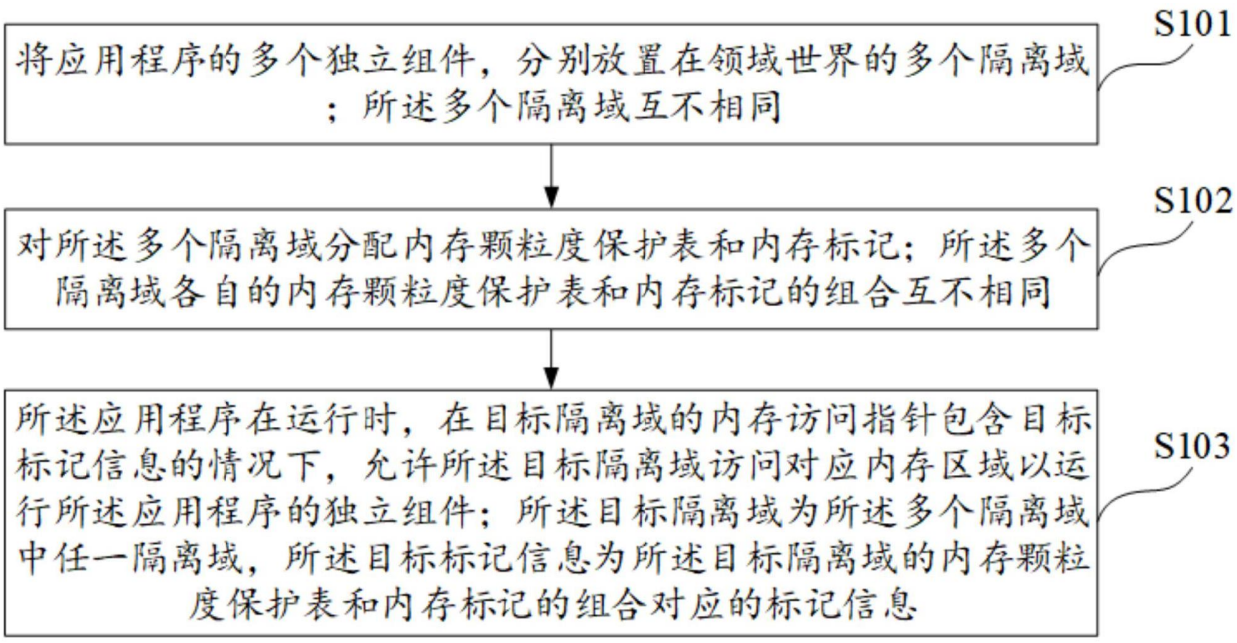

S101
将应用程序的多个独立组件，分别放置在领域世界的多个隔离域；所述多个隔离域互不相同
S102
对所述多个隔离域分配内存颗粒度保护表和内存标记；所述多个隔离域各自的内存颗粒度保护表和内存标记的组合互不相同
S103
所述应用程序在运行时，在目标隔离域的内存访问指针包含目标标记信息的情况下，允许所述目标隔离域访问对应内存区域以运行所述应用程序的独立组件；所述目标隔离域为所述多个隔离域中任一隔离域，所述目标标记信息为所述目标隔离域的内存颗粒度保护表和内存标记的组合对应的标记信息

图1

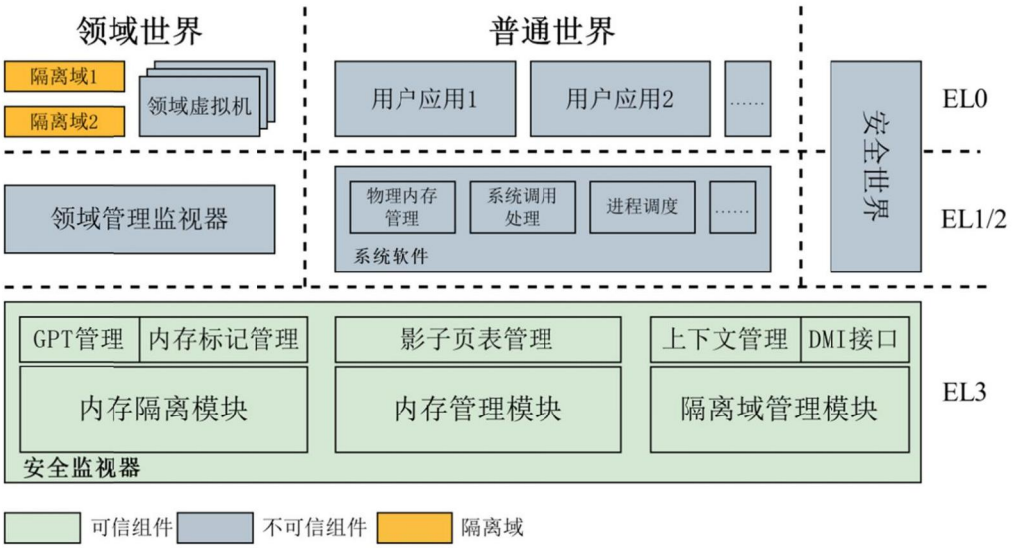

领域世界
普通世界
隔离域1
隔离域2
领域虚拟机
用户应用1
用户应用2
.....
安全世界
EL0
领域管理监视器
物理内存管理
系统调用处理
进程调度
.....
系统软件
EL1/2
GPT管理
内存标记管理
影子页表管理
上下文管理
DMI接口
内存隔离模块
内存管理模块
隔离域管理模块
EL3
安全监视器
可信组件
不可信组件
隔离域

图2

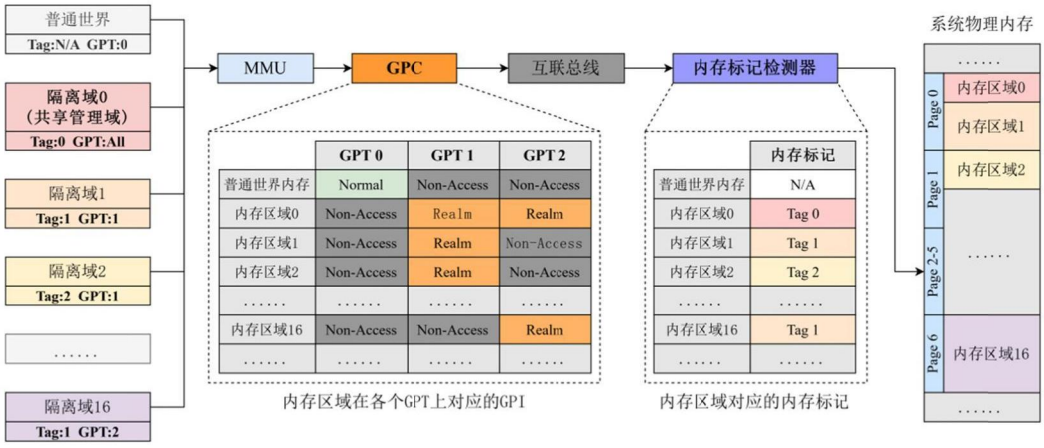

普通世界
Tag:N/A GPT:0
隔离域0
（共享管理域）
Tag:0 GPT:All
隔离域1
Tag:1 GPT:1
隔离域2
Tag:2 GPT:1
隔离域16
Tag:1 GPT:2
MMU
GPC
互联总线
内存标记检测器
GPT 0
GPT 1
GPT 2
普通世界内存 Normal Non-Access Non-Access
内存区域0 Non-Access Realm Realm
内存区域1 Non-Access Realm Non-Access
内存区域2 Non-Access Realm Non-Access
内存区域16 Non-Access Non-Access Realm
内存区域在各个GPT上对应的GPI
内存标记
普通世界内存 N/A
内存区域0 Tag 0
内存区域1 Tag 1
内存区域2 Tag 2
内存区域16 Tag 1
内存区域对应的内存标记
系统物理内存
Page 0
Page 1
Page 2-5
Page 6
内存区域0
内存区域1
内存区域2
内存区域16

图3

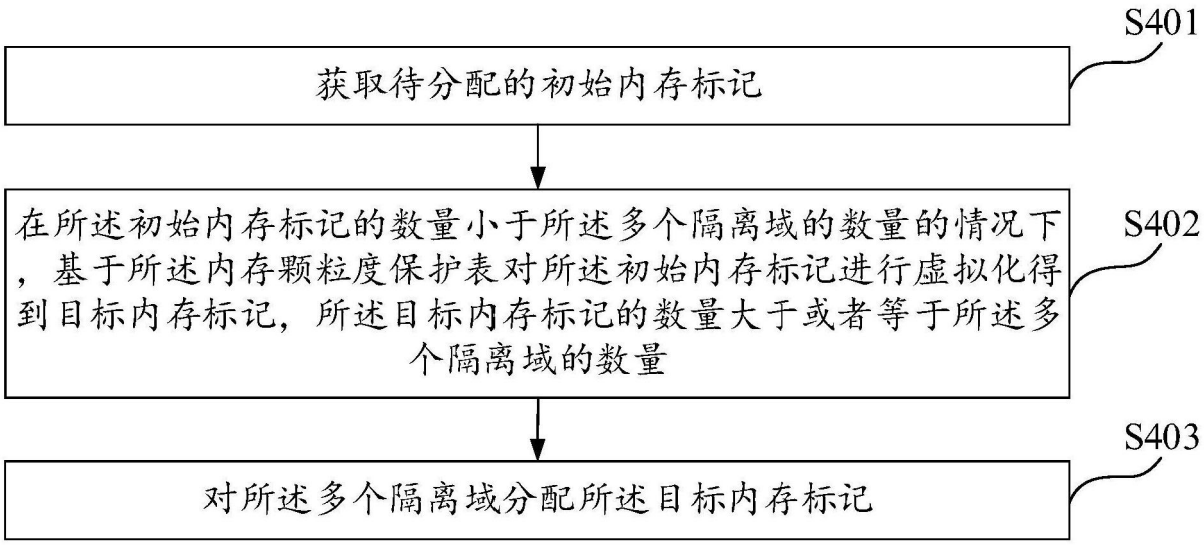

S401
获取待分配的初始内存标记
S402
在所述初始内存标记的数量小于所述多个隔离域的数量的情况下，基于所述内存颗粒度保护表对所述初始内存标记进行虚拟化得到目标内存标记，所述目标内存标记的数量大于或者等于所述多个隔离域的数量
S403
对所述多个隔离域分配所述目标内存标记

图4

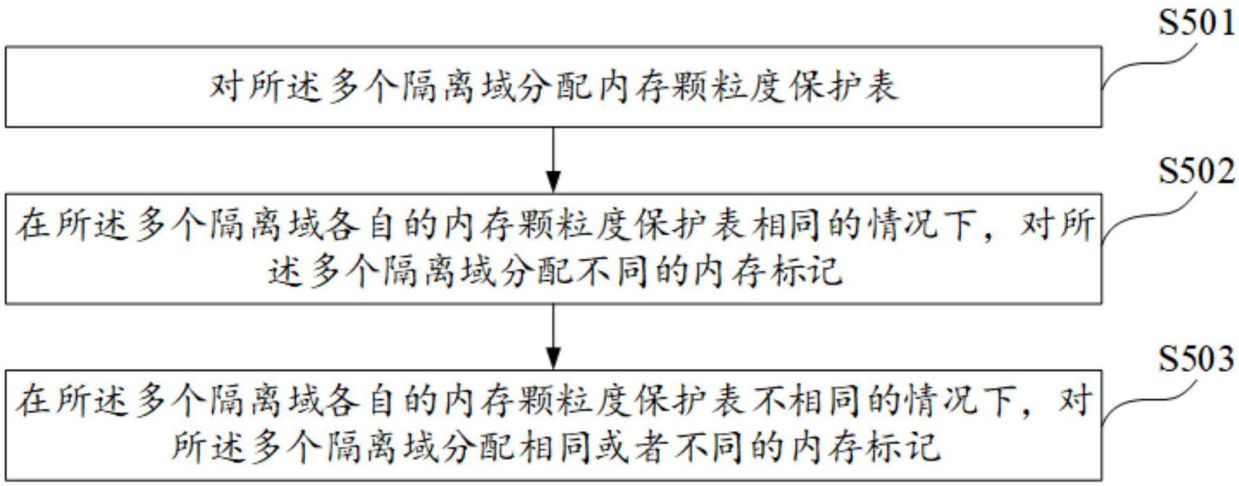

S501
对所述多个隔离域分配内存颗粒度保护表
S502
在所述多个隔离域各自的内存颗粒度保护表相同的情况下，对所述多个隔离域分配不同的内存标记
S503
在所述多个隔离域各自的内存颗粒度保护表不相同的情况下，对所述多个隔离域分配相同或者不同的内存标记

图5

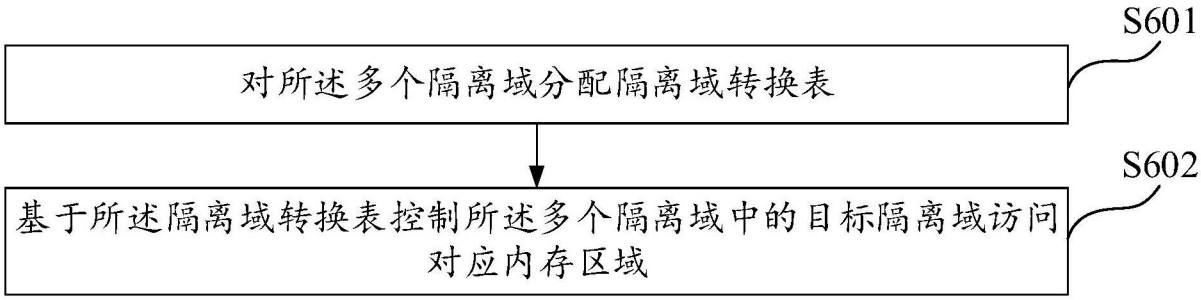

S601
对所述多个隔离域分配隔离域转换表
S602
基于所述隔离域转换表控制所述多个隔离域中的目标隔离域访问对应内存区域

图6

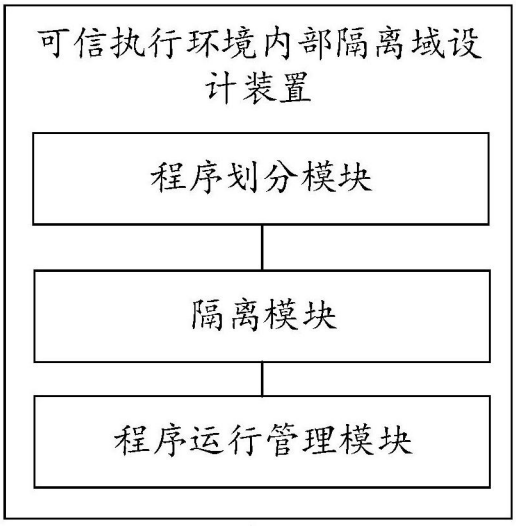

可信执行环境内部隔离域设计装置
程序划分模块
隔离模块
程序运行管理模块

图7

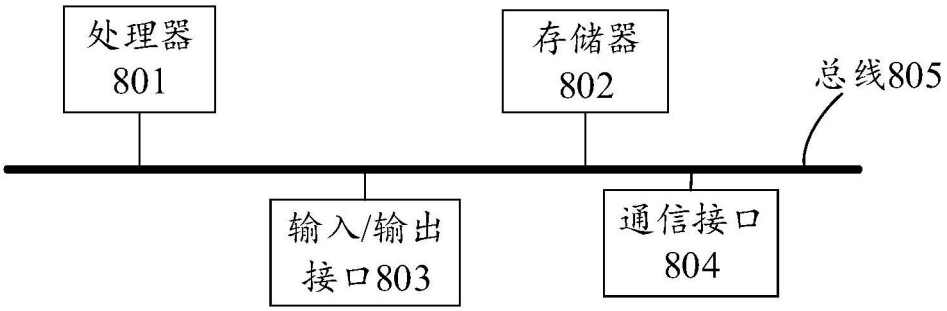


图8