(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 110598412 B
(45)授权公告日 2021.12.14

(21)申请号 201810599752.2

(22)申请日 2018.06.12

(65)同一申请的已公布的文献号
申请公布号 CN 110598412 A

(43)申请公布日 2019.12.20

(73)专利权人 杨力祥
地址 100081 北京市海淀区广源闸紫竹院公园宿舍1号楼3门101号

(72)发明人 杨力祥

(51)Int.Cl.
G06F 21/57(2013.01)
G06F 21/52(2013.01)

(56)对比文件
CN 105787477 A,2016.07.20
CN 105787477 A,2016.07.20
CN 103188249 A,2013.07.03
CN 102970414 A,2013.03.13
CN 103312801 A,2013.09.18
CN 106778291 A,2017.05.31
CN 104951410 A,2015.09.30
CN 206532131 U,2017.09.29
CN 106304040 A,2017.01.04
CN 101073059 A,2007.11.14
US 2005201188 A1,2005.09.15
CN 102970414 A,2013.03.13

审查员 张瑀琪

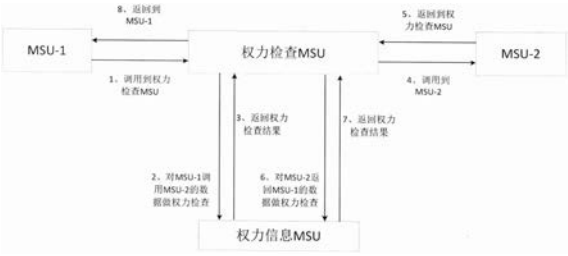
权利要求书2页 说明书13页 附图1页

(54)发明名称

将权力信息隔离并依托它进行权力检查的方法及计算装置

(57)摘要

本发明公开了一种访问控制方法，涉及信息技术，特别是信息安全领域，包括：确保权力信息的正确性以及依托权力信息进行权力检查。所述确保权力信息的正确性包括：在内存空间上，将权力信息及维护它的代码与软件系统其余部分隔离；在任何一个权力信息的计算处理过程中，不再与外界做交互。所述依托权力信息进行权力检查包括：在用户指定任务完成过程中，在指定位置处依据权力信息对软件系统处理的数据进行检查。应用本发明提供的方案，权力信息不会被攻击者篡改，也无法获得有效的攻击结果。



CN 110598412 B

1.一种将权力信息隔离并依托权力信息进行权力检查的方法,其特征在于:将权力信息及维护权力信息的代码与软件系统其余部分隔离以确保权力信息的正确性,以及依托权力信息进行检查;所述权力信息,包括:用户信息和用户对文件读写范围;

在任何一个权力信息的计算处理过程中,不再与外界做交互,包括:在初始化阶段,通过启动程序,引发一段权力信息加载专用程序执行,将所有权力信息一次性全部载入权力信息MSU中;关机前,由内核通用关机程序,引发一段权力信息同步专用程序将权力信息全部同步到外设上,确保外设上的存储的权力信息与内存中权力信息一致;如果权力信息MSU接收到新建文件请求,则内部的文件权力信息处理程序分析文件路径,并最终添加文件管理结构信息;如果接收到删除文件请求,则分析文件路径,并最终删除文件管理结构信息;如果接收到修改文件名的请求,则分析文件路径,并最终更改文件名对应目录项中内容;如果权力信息MSU接收到写文件请求,则内部的数据块处理程序通过文件管理结构找到数据块索引信息,并在索引信息管理结构中添加数据块号,如果接收到删除逻辑块请求,则通过文件管理结构找到数据块索引信息,并在索引信息管理结构中删除数据块号;MSU中的文件管理信息处理专用程序和数据块处理专用程序,通过自身就可以完成指定的权力信息处理工作,不需要任何外界的支持。

2.根据权利要求1所述的方法,其特征在于:所述将权力信息及维护权力信息的代码与软件系统其余部分隔离以确保权力信息的正确性,包括:在内存空间上,将权力信息及维护权力信息的代码与软件系统其余部分隔离。

3.根据权利要求1-2之一所述的方法,其特征在于:所述依托权力信息进行权力检查,包括:在用户指定任务完成过程中,在指定位置处依据权力信息对软件系统处理的数据进行检查。

4.根据权利要求3所述的方法,其特征在于:在内存空间上,将权力信息及维护权力信息的代码与软件系统其余部分隔离,分别称作权力封装和其他封装,包括:在同一个线性地址空间内,将权力信息及维护权力信息的程序独立的封装并与软件的其余部分程序分开存储。

5.根据权利要求4所述的方法,其特征在于:所述其他封装中包括检查封装,所述检查封装是指将执行检查功能的程序进行单独的封装,根据权力检查结果决定程序后续执行。

6.根据权利要求4-5之一所述的方法,其特征在于:将所述封装用MSU实现,所述MSU是指内存系统单元。

7.根据权利要求3所述的方法,其特征在于:所述在用户指定任务完成过程中,在指定位置处依据权力信息对软件系统处理的数据进行检查,包括:将与权力信息及其维护代码无关的程序,封装成不同的普通MSU,普通MSU之间不能直接调用或返回,而是要先进行权力检查。

8.根据权利要求7所述的方法,其特征在于:由普通MSU调用到权力检查MSU,权力检查MSU再调用到权力信息MSU,并将与权力相关的数据,传递给权力信息MSU做比较,权力信息MSU会将检查结果返回给权力检查MSU,如果比较结果超出用户权力信息限定的范围,进入异常处理流程,如果没有超出范围,再由权力检查MSU调用到目标普通MSU去执行;和/或,由普通MSU返回到权力检查MSU,权力检查MSU再调用到权力信息MSU,并将与权力相关的数据,传递给权力信息MSU做比较,将检查结果返回给权力检查MSU,如果比较结果超出用户权力

信息限定的范围，进入异常处理流程，如果没有超出范围，再由权力检查MSU返回到原普通MSU去执行。

9.一种访问控制计算装置，其特征在于：使用权利要求1-8之一的方法。

10.一种安全操作系统，其特征在于：使用权利要求1-8之一的方法。

将权力信息隔离并依托它进行权力检查的方法及计算装置

技术领域

[0001] 本申请涉及信息技术领域，特别涉及一种访问控制技术，以及一种通过权力控制防止攻击的技术。

背景技术

[0002] 现有技术中，软件难免存在设计缺陷，这些缺陷又往往成为可供攻击利用的“漏洞”。例如，利用超越范围的数组拷贝，数组下标操作导致的数组越界等方法，攻击者可以用准备好的数据修改内核的数据、代码，进而发起攻击。

[0003] 在此基础上，攻击者就可以进一步修改授权信息、改变授权状态，从而获得超越授权的状态。进而可以进行诸如以下操作：

[0004] 1、超越授权读取用户数据(包括内存和外设的数据)。

[0005] 2、超越授权写入(包括篡改、删除)用户数据。

[0006] 3、超越授权执行系统调用。

[0007] 4、超越授权执行应用程序。

[0008] 产生上述问题的根源之一，在于目前冯.诺依曼架构下的内存结构是几乎平坦的，因此一旦发生攻击，攻击者就可能跳转到它想要的几乎任意目标位置，进而完成数据覆盖等操作，从而获得超越授权的状态。进一步的，操作系统中对于用户的访问控制，又缺乏有效的机制，导致攻击程序很容易通过攻击获得超越授权，并保持这种状态。

[0009] 特别是对于类似Dirty Cow这样的攻击，其利用操作系统内核的竞争条件，通过可以称作“外部震荡”的方式，引发竞争，进而取得超越授权的效果。

发明内容

[0010] 针对现有技术的问题，本发明建立一种访问控制方法，其特征在于：将权力信息及维护它的代码与软件系统其余部分隔离以确保权力信息的正确性，以及依托权力信息进行检查。

[0011] 优选的，所述检查是指在系统的执行过程中，跨MSU访问时，依据隔离的权力信息进行权力检查。

[0012] 所述权力信息包括：用户信息和用户对文件读写范围。例如，用来表示文件所属用户的信息；文件所属用户组的信息；用户、用户组、组外用户、对文件的读、写、执行权限信息；文件数据块号索引信息等。

[0013] 所述将权力信息及维护它的代码与软件系统其余部分隔离以确保权力信息的正确性包括：在内存空间上，将权力信息及维护它的代码与软件系统其余部分隔离；和/或，在任何一个权力信息的计算处理过程中，不再与外界做交互。所述依托权力信息进行权力检查包括：在用户指定任务完成过程中，在指定位置处依据权力信息对软件系统处理的数据进行检查。

[0014] 所述在内存空间上，将权力信息及维护它的代码与软件系统其余部分隔离，包括：

在同一个线性地址空间内，将权力信息及维护它的程序独立的封装并与软件的其余部分程序分开存储。优选的，用终端MSU存储权力信息及维护它的代码(此类MSU以后简称权力信息MSU)。进一步包括：将执行检查功能的程序进行单独的封装，根据权力检查结果决定程序后续执行，称为检查封装，优选的，利用检查MSU实现检查封装，(此类MSU以后简称为权力检查MSU)。

[0015] 所述MSU是指内存系统单元，所述内存系统单元是内存系统装置中的某个具体单元；所述内存系统装置是指特定访问控制的集合及其控制的访问区域。

[0016] 除非特别指明，本发明中MSU这一缩写对应的就是内存系统单元(Memory System Unit)。

[0017] 所述区域，包括：由一组边界包围而成的CPU可寻址存储空间，区域必须由访问控制集合认定，所述认定是指将区域的信息记录在MSU信息中。所述访问控制集合，包括：MSU信息，对区域进行访问的允许机制，和/或对区域进行访问的禁止机制。所述可寻址存储空间可以存放数据和/或指令。优选的，全部软件的数据、代码都按设计要求分别放入指定的MSU之中，即没有代码、数据放在MSU之外。

[0018] 所述CPU是指中央处理器。

[0019] 进一步，区域由同一个线性地址空间中的一个或多个连续存储区组成，每个连续存储区由两端的地址标识界定，所有前述的地址标识的集合构成区域的边界。对于由多个连续存储区组成的区域的优选方案是区域中的连续存储区之间互不相交。其中存储数据、代码的存储区分别被称作数据区、指令区。不同 MSU的区域互不相交。

[0020] 进一步的，所述MSU信息包括：MSU边界信息、MSU端口信息、MSU 属性信息。作为一种可选的实现方式，可以设置空端口MSU，所述空端口MSU 其MSU端口信息为空，仍具有MSU边界信息、MSU属性信息。

[0021] 优选的，所述MSU信息进一步包括：MSU用户信息。

[0022] 进一步的，所述允许机制包括：允许区域内的非转移指令、中断指令及目标地址在当前区域内(不超越当前区域)的转移指令执行，允许区域内的指令访问当前区域内的数据。进一步的，允许机制包括：允许区域间，不论是区域内到区域外或区域外到区域内，通过传参的方式传递数据；允许区域间通过共享物理内存的方式传递数据，优选的，传递大量的数据时采用共享物理内存的方式；对区域间，即超出或进入本区域，进行访问的允许机制，进一步包括： MSU间必须经过端口执行转移指令，并且属性信息、端口信息必须匹配。

[0023] 所述禁止机制包括，禁止在区域中的数据区执行指令。除允许机制之外，对一切由区域内向区域外或由区域外向区域内的跨区域执行指令(包括非转移指令、转移指令及不匹配情形)，跨区域操作访问数据都产生异常。

[0024] 一个特例是共享数据MSU，其特征是只包含被其他MSU共享的数据，没有指令；允许其他MSU通过约定的指令操作数据。

[0025] 在本发明的一种具体实现方式中，将内核栈和/或用户栈置于共享数据MSU 中，栈所属的MSU必须为共享数据MSU，其他MSU通过约定的指令操作栈中的数据。

[0026] 所述MSU边界信息包括：由访问控制集合认定的区域中，所有连续存储区的边界信息构成的集合。存储上述信息的数据结构简称边界数据，所述边界数据的地址被关联到内存系统装置中并为其可识别。当需要查找区域的边界时，所述装置可以根据边界数据的地

址找到数据结构,即可获得所有的边界信息。

[0027] 所述MSU端口信息包括入口和/或出口。在访问控制集合认定的区域范围内的指令地址区域中指定有限个指令地址为入口或出口,其中每一个指令地址为一个入口或出口。可选的入口为:区域中MSU间转移指令的目标地址;可选的出口为:MSU间转移指令的所在地址。

[0028] 所述MSU属性信息包括:MSU标识信息,MSU类型信息。所述MSU标识信息是指区别于其它MSU的唯一标识。所述MSU的类型信息可以是普通MSU、共享数据MSU中的一种。

[0029] 优选的,所述MSU属性信息还可包括:MSU所属用户类型信息,MSU所属用户标识信息。所述MSU所属用户类型信息是指这个MSU所属用户的类型,在一些应用场景中,用户类型即为用户角色,所述MSU所属用户标识信息是指 MSU所属用户的唯一标识。

[0030] 优选的,可以将前述的边界信息和/或属性信息和/或MSU端口信息合成为一个更方便使用的、完整的数据结构。

[0031] 所述MSU端口信息匹配、所述MSU属性信息匹配是指:在程序初始化阶段,将转移指令执行所需MSU的出口、入口、边界、标识信息、类型信息记录在MSU描述符表中,在程序运行时,将转移指令包含的信息,分别与MSU描述符表中的端口信息、属性信息做对比,如果结果匹配,视为合法,允许转移指令执行,反之,视为非法,报异常。

[0032] 进一步的,在MSU类型信息中增加一种检查MSU。类型信息被标记为“检查MSU”的MSU被视为检查MSU。当所述装置包含检查MSU时,不允许非检查MSU直接调用另外一个非检查MSU,必须由源MSU先调用检查MSU,再由检查MSU调用目标MSU;目标MSU返回时,先返回到检查MSU,再由检查MSU返回到源MSU。所述非检查MSU指除了检查MSU外的任何其它类型的MSU。

[0033] 进一步的,在MSU类型信息中增加一种终端MSU。类型信息标记为“终端MSU”的MSU只可被其它MSU调用,不可调用其它MSU。

[0034] 进一步的,在MSU类型信息中增加一种空端口MSU。类型信息被标记为“空端口MSU”的MSU没有端口,其它MSU可以通过端口调用任意空端口 MSU的函数,但不可直接访问空端口MSU的数据。空端口MSU调用其它MSU 必须通过其端口进入该MSU。不同的空端口MSU之间可以任意进行函数调用,但不可访问数据。当终端MSU存在时,空端口MSU不可调用终端MSU。

[0035] 进一步的,在MSU类型信息中增加一种保险箱MSU。此类MSU不允许包含指令区。只有某些需要保存状态信息的操作,才可访问该MSU。优选的,所述状态信息可以是返回地址、中断现场等。

[0036] 进一步的,在MSU类型信息中增加一种IO指令MSU。当所述装置包含IO 指令MSU时,仅允许这类MSU内执行IO操作相关的特殊指令。此类MSU的属性匹配检查规则与终端MSU相同。

[0037] 在装置中,可不支持检查MSU、终端MSU、空端口MSU、保险箱MSU、 IO指令MSU的实现,也可支持其中的一种或几种。

[0038] 所述在任何一个权力信息的计算处理过程中,不再与外界做交互,包括:在初始化阶段,通过启动程序,引发一段权力信息加载专用程序执行,将所有权力信息一次性全部载入权力信息MSU中;关机前,由内核通用关机程序,引发一段权力信息同步专用程序将权力信息全部同步到外设上,确保外设上的存储的权力信息与内存中权力信息一致;如果权力

信息MSU接收到新建文件请求,则内部的文件权力信息处理程序分析文件路径,并最终添加文件管理结构信息;如果接收到删除文件请求,则分析文件路径,并最终删除文件管理结构信息;如果接收到修改文件名的请求,则分析文件路径,并最终更改文件名对应目录项中内容;如果权力信息MSU接收到写文件请求,则内部的数据块处理程序通过文件管理结构找到数据块索引信息,并在索引信息管理结构中添加数据块号,如果接收到删除逻辑块请求,则通过文件管理结构找到数据块索引信息,并在索引信息管理结构中删除数据块号;MSU中的文件管理信息处理专用程序和数据块处理专用程序,通过自身就可以完成指定的权力信息处理工作,不需要任何外界的支持。

[0039] 所述在用户指定任务完成过程中,在指定位置处依据权力信息对软件系统处理的数据进行检查,包括:将与权力信息及其维护代码无关的程序,封装在不同的MSU内,此类MSU的属性为普通MSU(以后简称此类MSU为功能 MSU),功能MSU之间不能直接调用或返回,而是要先进行权力检查,当功能 MSU-1对功能MSU-2有调用需求的时候,一种优选的方式是:以下结合附图1 进行说明,在源代码中记录着MSU-1对MSU-2的调用需求,实际执行时,由功能MSU-1调用到权力检查MSU(如图1步骤1),权力检查MSU再调用到权力信息MSU(如图1步骤2),并将与权力相关的数据,传递给权力信息MSU 做比较,权力信息MSU会将检查结果返回给权力检查MSU(如图1步骤3),如果比较结果超出用户权力信息限定的范围,进入异常处理流程,如果没有超出范围,再由权力检查MSU根据调用需求,实际调用到目标功能MSU-2去执行(如图1步骤4);和/或,当功能MSU-2返回时,先返回到权力检查MSU(如图1步骤5),权力检查MSU再调用到权力信息MSU(如图1步骤6),并将与权力相关的数据,传递给权力信息MSU做比较,它将检查结果返回给权力检查 MSU(如图1步骤7),如果比较结果超出用户权力信息限定的范围,进入异常处理流程,如果没有超出范围,再由权力检查MSU返回到功能MSU-1去执行 (如图1步骤8)。

[0040] 一种访问控制方法,其特征在于:

[0041] 确保MSU中内容功能单一,包括:确保每一个MSU中的内容,只能完成用户指定任务中的一部分功能,这部分功能无法单独实现越权。

[0042] 一种访问控制机制,其特征在于:使用前述的基于权力进行访问控制的方法。

[0043] 一种安全操作系统,其特征在于:使用前述的基于权力进行访问控制的方法。

[0044] 通过上述方法,本发明能够起到以下技术效果:

[0045] 权力信息是权力检查的基础,将权力信息及其维护程序与软件系统其余部分,在空间上隔离,特别是通过MSU机制实现隔离,可以有效的避免由于软件系统其余部分内容被攻击而影响到权力信息的正确性。在保证正确性的基础上,每当进行MSU间访问时,依据隔离的权力信息进行权力检查,可以进一步避免 MSU间的越权操作。

[0046] 在任何一个权力信息的计算处理过程中,不再与外界做交互,首先可以使权力信息的处理变得极为简单且功能单一,通过形式化测试和穷举测试,就可以保证其正确性;其次由于在处理过程中不需要与外界做交互,也就保证了权力信息处理过程不受外界影响,保证了权力信息处理的正确性。

[0047] 将软件系统其余部分进行独立封装,特别是通过MSU机制进行封装,可以保证每个MSU中的程序只能通过有限的端口进行调用和返回,在端口处进行权力检查,可以保证MSU间交互时,必须经过权力检查,以此实现MSU间无法越权操作。在所有MSU间进行权力检查,

可以保证在整个程序执行过程中,都无法实现MSU间的越权操作。

[0048] 越权操作需要程序中多个功能协同配合才能实现,通过MSU机制,确保每个MSU功能单一,可以确保MSU内的程序无法独立实现越权。

[0049] 将对外设的操作或对用户数据的操作封装在终端MSU中,并确保此类终端 MSU中不包含其余内容,而且其余MSU如果需要与外设进行交互或操作用户数据,只能通过此类终端MSU。这样做可以避免其它MSU中程序通过与外设直接交互或操作用户数据的方式,直接实现越权。

附图说明

[0050] 为了更清楚地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

[0051] 图1:MSU-1调用MSU-2必须通过权力检查MSU进行权力检查的示意图

具体实施方式

[0052] 下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清楚、完整地描述,显然,所描述的实施例仅仅是本发明一部分实施例,而不是全部的实施例。基于本发明中的实施例,本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例,都属于本发明保护的范围。

[0053] 以下通过具体实施方式来进一步说明本发明的技术内容。

[0054] 实施例1

[0055] 对于“在任何一个权力信息的计算处理过程中,不再与外界做交互”的一种实施方式是:

[0056] 系统启动后,所有的普通MSU中内容加载前,通过启动程序,引发一段权力信息加载的专用程序执行,将所有权力信息一次性全部载入权力信息封装中,权力信息加载的专用程序中只包括加载权力信息的逻辑,除此之外没有其它内容,以此确保其单一且逻辑简单,通过形式化测试和穷举测试就可以确定此专用程序的正确性,加载时它自身不会产生攻击而影响到权力信息的可靠性,同时,由于软件系统处理的数据还没有加载,所以此时还没有其它程序执行,也不会产生攻击,这样已有的权力信息和权力信息处理程序载入后,也是正确的。

[0057] 实施例2

[0058] 对于“在任何一个权力信息的计算处理过程中,不再与外界做交互”的一种实施方式是:

[0059] 关机前,由内核通用关机程序,引发一段权力信息同步的专用程序将权力信息全部同步到外设上,确保外设上的存储的权力信息与内存中权力信息一致。此专用程序的逻辑,包括往外设上同步权力信息,除此之外,没有其它内容,以此确保其功能单一且逻辑简单,通过形式化测试和穷举测试就可以确定此专用程序的正确性,同步时它自身不会产生攻击而影响到权力信息的可靠性。

[0060] 实施例3

[0061] 对于“在任何一个权力信息的计算处理过程中，不再与外界做交互”的一种实施方式是：

[0062] 以创建文件为例：

[0063] 通过系统调用软中断，进入系统调用对应的功能MSU后，功能MSU会接收到包括需要创建文件的路径、读写属性、创建文件标记等参数，会先调用到权力检查MSU进行权限检查，权力检查MSU会将传递的参数，传递给权力信息MSU，具体的检查工作在权力信息MSU中，由文件权力信息处理专用程序进行，即通过分析路径名，确定当前用户是否有权力访问各级目录文件，如果检查通过，最后为目标文件创建添加文件管理结构(其中文件权力属性字段中包含用户身份信息、用户组信息，以及用户自身、用户组、组外用户对新建文件的“读”、“写”、“执行”权限)，并在文件属性管理结构位图上做标识；同时在文件操作管理结构、文件句柄对应的结构中，找到空闲表项，并通过表项建立文件操作管理结构——文件句柄对应结构——文件属性管理结构的对应关系。整个完成工作不需要与外界进行交互。

[0064] 以删除文件为例：

[0065] 通过系统调用软中断，进入系统调用对应的功能MSU后，功能MSU会接收到包括需要删除文件的路径等参数，会先调用到权力检查MSU进行权限检查，权力检查MSU会将传递的参数，传递给权力信息MSU，具体的检查工作在权力信息MSU中，由文件权力信息处理专用程序进行，即通过分析路径名，确定当前用户是否有权力访问各级目录文件，如果检查通过，最后为删除目标文件对应的文件管理结构，并在文件管理结构位图上将与此文件管理结构对应的位置0，以及将相应目录文件中与把此文件管理结构对应的目录项清空。整个完成工作不需要与外界进行交互。

[0066] 以更改文件名为例：

[0067] 通过系统调用软中断，进入系统调用对应的功能MSU后，功能MSU会接收到包括需要更改文件的路径、需要改写成的文件名等参数，会先调用到权力检查MSU进行权限检查，权力检查MSU会将传递的参数，传递给权力信息 MSU，具体的检查工作在权力信息MSU中，由文件权力信息处理专用程序进行，即通过分析路径名，确定当前用户是否有权力访问各级目录文件，如果检查通过，最后在相应的目录文件中，将相应目录项中文件名内容改写为指定文件名。整个完成工作不需要与外界进行交互。

[0068] 实施例4

[0069] 对于“在任何一个权力信息的计算处理过程中，不再与外界做交互”的一种实施方式是：

[0070] 以向文件中写入内容为例：

[0071] 通过系统调用软中断，进入系统调用对应的功能MSU后，会接收到包括写入目标文件的文件句柄、需要写入数据在进程空间的地址、写入的字节数在内的参数，会先调用到权力检查MSU进行权限检查，权力检查MSU会将传递的参数，传递给权力信息MSU，具体的检查工作在权力信息MSU中，由数据块处理专用程序进行，通过文件句柄，获取文件文件管理结构，检查此文件管理结构是否处于当前用户可操作文件范围内，如果检查通过，进一步获取数据块索引信息，通过文件偏移标记和参数中的写入字节数，确定待写入数据需要占用的数据块数量以及在文件中的逻辑位置；通过数据块管理位图，确定需要占用块设备上哪些

空闲数据块，确定块号，并最终将数据块号写入索引信息中。整个完成工作不需要与外界进行交互。

[0072] 以删除文件中内容为例：

[0073] 通过系统调用软中断，进入系统调用对应的功能MSU后，会接收到包括将要删除目标文件的文件句柄、要保留的文件内容大小等参数，会先调用到权力检查MSU进行权限检查，权力检查MSU会将传递的参数，传递给权力信息 MSU，具体的检查工作在权力信息MSU中，由数据块处理专用程序进行，通过文件句柄，获取文件文件管理结构，检查此文件管理结构是否处于当前用户可操作文件范围内，如果检查通过，根据保留的文件内容大小，改写文件管理结构中标识文件大小的字段以调整文件大小。整个完成工作不需要与外界进行交互。

[0074] 实施例5

[0075] 对于“利用MSU的特性，在除权力信息及维护它的代码之外的内容所在 MSU之间，进行权力检查”的一种实施方式是：

[0076] 功能MSU中只包含功能信息、对功能信息进行处理的程序以及对外部进行访问的逻辑；功能MSU只能通过指定的端口才能对外部访问，而且只能调用或返回到权力检查MSU，权力检查MSU对其所传递数据中与权力相关的数据做检查，检查通过后，再由权力检查MSU调用或返回到其它功能MSU端口函数；检查MSU只负责接收功能MSU传递的数据，并根据检查的结果是否通过，决定是继续完成功能MSU间的访问，还是进入异常处理流程，对于实际的权力检查工作，权力检查MSU会调用到权力信息MSU，再由权力信息MSU用下传的权力信息和已有的权力信息作比对，确定用户是否越权，并将比对结果返回给权力检查MSU。

[0077] 为此需要在功能MSU中，针对权力检查，需要添加指令，逻辑是：

[0078] 功能MSU端口函数调用权力检查MSU端口函数前，需要添加指令的逻辑包括：

[0079] 传递参数(参数中包含权力信息)

[0080] 传递当前功能MSU的ID号

[0081] 传递当前功能MSU出口号

[0082] 传递目标功能MSU的ID号

[0083] 传递目标功能MSU入口号。

[0084] 权力检查MSU中，针对权力检查，需要添加指令的逻辑包括：

[0085] 根据原功能MSU的ID号、原MSU出口号、目标功能MSU的ID号、目标MSU入口号，找到对应的检查逻辑

[0086] 从参数中提取权力信息，并调用相应的权力信息MSU端口函数，将权力信息以参数形式传递给权力信息MSU

[0087] 接收权力信息MSU返回的权力检查结果，如果权力检查没通过，进入异常处理流程，如果权力检查通过，调用目标功能MSU的端口函数

[0088] 从目标功能MSU端口函数的返回值中提取权力信息，并调用相应的权力信息MSU端口函数，将权力信息以参数形式传递给权力信息MSU

[0089] 接收权力信息MSU返回的权力检查结果，如果权力检查没通过，进入异常处理流程，如果权力检查通过，返回到原功能MSU的端口函数，并将返回值回传给原功能MSU。

[0090] 一个具体的应用是：

[0091] 读文件过程中进行的权力检查：

[0092] 步骤1：通过系统调用软中断，进入系统调用对应的功能MSU后，会接收到包括读取目标文件的文件句柄、需要读出数据存储在进程空间的地址、读取的字节数，之后调用读文件功能对应的功能MSU前，需要先调用到权力检查 MSU进行权限检查，权力检查MSU会将传递的参数，传递给权力信息MSU，具体的检查工作在权力信息MSU中进行，权力信息MSU接到参数后，通过分析文件句柄，确定此次读取的文件处于当前用户能够访问的所有文件范围内；之后再对参数“读取的字节数以及文件偏移”进行权力检查，通过它们的数值，可以精确的圈定此次访问文件的数据块范围，还可以确定此次要访问的数据块，哪些已经被载入到缓冲区，哪些还没有，这些权力信息，将作为后续MSU中，对数据块进行权力一致性检查的依据。

[0093] 步骤2：读文件功能对应的功能MSU调用缓冲区处理功能对应的功能MSU 时，也要通过权力检查MSU——权力信息MSU，先进行权力检查，检查文件号与此次要读取的目标文件的文件号是否一致，如果不一致，便视为越权，进入异常处理流程。

[0094] 步骤3：缓冲区处理功能对应的功能MSU调用文件管理功能对应的MSU 时，也要通过权力检查MSU——权力信息MSU，先进行权力检查，检查要操作的数据块是否属于当前要读取的文件，由于系统调用对应的功能MSU调用读文件功能对应的功能MSU前进行权力检查时，就已经通过读取字节数和文件偏移这两个参数精确圈定了此次要读取的数据块范围，所以此时检查数据块的文件归属时，不仅能检查出数据块是否属于要读取的目标文件(如果不属于目标文件，即被视为越权)，而且还能检查出是否属于本次要操作的数据块范围内，如果超出，即便不是越权，也可以做出提示，进入异常处理流程。

[0095] 步骤4：如果缓冲区处理功能对应的功能MSU确定要读取的数据块没有载入缓冲区，还会调用页面处理功能对应的功能MSU，准备申请缓冲块，也要通过权力检查MSU——权力信息MSU，先进行权力检查，检查申请到的页面是否属于其它用户，如果属于，视为越权，进入异常处理流程。

[0096] 步骤5：申请到缓冲块后，缓冲区处理功能对应的功能MSU调用请求项处理功能对应的功能MSU，也要通过权力检查MSU——权力信息MSU，先进行权力检查，检查下传的缓冲块号、设备号、块号、缓冲块所在页面号，是否对应着当前要访问的文件，如果不一致，就视为越权，进入异常处理流程。

[0097] 步骤6：请求项处理功能对应的功能MSU调用驱动处理功能对应的功能 MSU，也要通过权力检查MSU——权力信息MSU，先进行权力检查，检查下传的绝对扇区号、读取扇区数，与本次指定的要读取的文件数据块和字节数是否匹配，如果不匹配，视为越权，进入异常处理流程。

[0098] 步骤7：驱动处理功能对应的功能MSU调用DMA命令发送对应的终端 MSU，调用前先进行权力检查，检查下传的DMA参数是否与此次要操作的数据块相匹配，如果不匹配，视为越权，进入异常处理流程，如果匹配，说明下达的数据操作命令没有越权，进入DMA命令发送对应的终端MSU，它是最后一步，直接发送DMA读盘命令。

[0099] 实施例6

[0100] 针对“确保MSU中内容功能单一”的一个实施方式是：

[0101] 比如往进程页面中写入数据，至少需要两个步骤，一步是找到指定的进程页面，另

一步是往指定页面中写入数据，为了确保功能单一，把找到指定的进程页面这部分内容，封装在一个普通MSU中，把往指定页面中写入数据这部分内容封装在一个终端MSU中，负责查找指定页面的MSU，不能写入数据；负责写入数据的MSU，不能指定页面，单独哪个MSU中的内容，都无法实现越权，而负责找到指定进程页面的MSU，在找到指定页面后，要先经过权力检查 MSU，检查当前用户是否有权力往指定的页面中写入数据，检查通过，再由权力检查MSU调用数据写入对应的MSU去执行，检查如果不通过，就会被拦截。

[0102] 实施例7：

[0103] 一种现有体系下通过软件指令进行访问控制的MSU制作方法及针对该方法的访问控制应用方式：

[0104] A1内存系统装置的制作，具体包括：

[0105] A1-1制作MSU信息记录单元：

[0106] 建立以下数据：

[0107] 当前MSU的ID；MSU控制对照表；端口匹配表；指向MSU控制对照表的指针变量；指向端口匹配表的指针变量；用以记录MSU栈底地址值的变量。

[0108] 所述MSU控制对照表的信息包括：所有MSU的信息，具体包括：MSU 的ID号、MSU的边界信息、属性信息、端口信息、生效/失效信息。优选的，还包括MSU所属用户类型信息，MSU所属用户标识信息。

[0109] 所述MSU边界信息，包括：指令区边界信息、全局数据区边界信息、堆区边界信息。

[0110] 所述MSU端口信息包括：MSU的出口信息和MSU的入口信；

[0111] 所述MSU的出口信息，包括：其所属的MSU的ID、出口号、出口地址值；所述MSU的入口信息，包括：其所属的MSU的ID、入口号、入口地址值；

[0112] 所述端口匹配表，包括：一对有MSU间调用关系的出口和入口。

[0113] 在每个MSU的数据区，设置：指向MSU控制对照表的指针变量；指向端口匹配表的指针变量；记录MSU栈底地址值的变量。

[0114] 在每个MSU数据区的线性地址空间中，以页对齐的方式预留一段空间，空间大小为页大小的整数倍，将控制对照表设置于其中，其它数据不存入其中。

[0115] A1-2制作访问控制单元

[0116] 在本制作方法中：MSU访问控制逻辑靠软件指令进行控制，具体包括：

[0117] ●获取当前MSU栈底地址值：

[0118] 添加指令的逻辑是：在MSU间调用的参数传递指令前，获取栈顶地址值，并将此地址值压入栈中，此地址值作为目标MSU的栈底地址值；调用进目标MSU 后，在其指令的起始位置，获取栈中传递的上述地址值，保存到用于记录当前 MSU栈底地址值的变量中。

[0119] ●添加检查指令用以判定数据访问是否超出MSU边界：

[0120] 由于对于非指针变量，可以在编译阶段明确访问地址，所以一种优选的方案是，运行时不再对它们进行边界判断，只需对数据指针进行边界检查，具体方式：在访问数据指针对应的指令之前，添加判断逻辑，来进行访问的边界检查，具体包括：

[0121] 步骤1：如果访问的最终目标地址处于当前MSU的全局数据区，或堆区，或处于栈区中当前MSU对应的区域内，跳转到步骤2，否则跳转到步骤3；

[0122] 步骤2：执行数据访问指令，跳转到步骤4；

[0123] 步骤3:进入异常处理流程;

[0124] 步骤4:执行下一条指令

[0125] ●添加检查指令用以判断MSU内间接转移指令的目标地址是否超出MSU 边界:

[0126] 由于对于MSU内的直接转移指令,可以在编译阶段明确转移目标地址,所以一种优选的方案是,运行时不再对它们进行边界判断,只需对MSU内间接转移指令的目标地址进行边界检查,具体方式:在MSU内间接转移指令之前,添加判断逻辑,来进行访问的边界检查,具体包括:

[0127] 步骤1:如果访问的最终目标地址处于当前MSU的指令区内,跳转到步骤 2,否则跳转到步骤3;

[0128] 步骤2:执行MSU内间接转移指令,跳转到步骤4;

[0129] 步骤3:进入异常处理流程;

[0130] 步骤4:执行下一条指令

[0131] ●MSU属性匹配检查:

[0132] 根据编译器和链接器,将MSU间调用指令所在地址信息和目标地址信息予以记载,并体现到检查指令中。

[0133] 根据MSU间调用指令的目标地址值和所有MSU的边界信息,确定目标 MSU,并进一步用当前MSU的属性和目标MSU的属性做比对,如果属性匹配符合发明内容中记载的MSU属性匹配规则,再进行端口匹配检查,否则,进入异常处理流程。

[0134] ●MSU端口匹配检查:

[0135] 端口检查的目的是:检查当前MSU调用、返回是否与预期的MSU间调用、返回一致,防止改变MSU间执行序。具体方式是:1,在MSU间调用前,检查当前调用指令的地址值与目标地址是否记录在端口匹配表中。2,在MSU间返回时,一个返回指令,可能对应多个合法的返回地址,如果进行出入口的匹配检查,可能导致执行效率降低,一种优选的方案是:在返回时,仅检查返回指令是否为合法的出口。

[0136] 在MSU间调用指令前添加逻辑如下:

[0137] 通过MSU间调用指令所在地址值,在端口匹配表中找到相应的出口,通过此出口,确定其匹配的入口;再判断MSU间调用指令目标地址值,是否与该入口地址值一致,如果一致,允许MSU间调用指令执行,否则,进入异常处理流程。

[0138] 在MSU间返回指令前添加逻辑如下:通过MSU间返回指令所在地址值,在当MSU控制对照表中找相应的出口,如果能够找到,说明这是一个合法的出口,允许MSU间返回指令执行,否则,进入异常处理流程。

[0139] ●对MSU中非转移指令和内部直接转移指令的检查:

[0140] 对于非转移指令,可通过编译确定其在所属MSU的区域范围内;对于内部直接转移指令,也可在编译阶段确保其目标地址在MSU的区域范围内。通过将指令区所在页面设置为只读,可保证指令在运行时不会被更改,为了提高执行效率,一种优选的方案是:依靠编译阶段保证其正确性,在运行时阶段不再对其进行检查。

[0141] ●对IO指令的检查:

[0142] 从语法树生成汇编指令时,在所有指定的IO指令前增加判断逻辑:判断当前MSU的类型是否为IO指令类型的MSU,如是,可继续执行,如不是,则报出异常。

[0143] 不论IO指令是高级代码生成还是直接嵌入的汇编,都需进行此操作,确保可执行程序中所有的IO指令前都包含此检查逻辑。

[0144] 所述IO指令为直接对外设进行读写的特殊指令,不同体系的CPU的IO指令各不相同,以实际为准,如INTEL体系下in、out指令。

[0145] 针对该种内存系统装置制作方式的访问控制应用方式,包括:

[0146] B1编译包含MSU的源程序,具体包括:

[0147] B1-1、提取MSU信息,具体包括:

[0148] B1-1-1:编写、编译包含MSU信息的源程序:

[0149] ●一种增设语法规则的方式表达MSU信息

[0150] 增设语法规则,使编程阶段完成准确保留程序设计中MSU信息,为了兼容性,本规则在C语言的基础上,增设如下语法规则:

[0151] MSU声明:

[0152] MSU类型 MSU名 生效/失效位

[0153] {

[0154] 数据声明

[0155] 访问标识符:函数声明

[0156] };

[0157] 访问标识符:inner、port

[0158] MSU类型:

[0159] common_msu

[0160] check_msu

[0161] terminal_msu

[0162] nothing_msu

[0163] share_msu

[0164] MSU空端口函数声明:

[0165] 返回值类型 MSU名 函数名 形式参数类型列表;

[0166] MSU空端口函数定义:

[0167] 返回值类型 MSU名 函数名 形式参数类型列表 复合语句

[0168] 端口函数声明:

[0169] 端口标识符 声明表 MSU名 函数名 形式参数类型列表;

[0170] 端口函数定义:

[0171] 声明表 MSU名 函数名 形式参数类型列表 复合语句

[0172] 端口函数调用:

[0173] 函数名 参数列表

[0174] 指针区域类型 指针定义

[0175] 指针区域类型:

[0176] data

[0177] stack

[0178] heap

[0179] 其中MSU类型代表MSU的属性:common_msu代表普通MSU、check_msu 代表检查MSU、terminal_msu代表终端MSU、nothing_msu代表空端口MSU、share_msu代表共享数据MSU。当MSU类型为空端口MSU时,不需要定义函数的访问标识符。

[0180] MSU名代表MSU的标识信息;一对{}里面的数据和函数,从属于同一个 MSU。

[0181] 由inner这个访问标识符标识的函数为MSU空端口函数;

[0182] 由port这个访问标识符标识的函数为MSU端口函数;

[0183] 生效/失效位,记录着MSU是否可用,1代表生效,0代表失效。

[0184] 共享数据MSU中只允许定义数据。

[0185] 指针区域类型:data标识的指针为全局数据区指针;stack标识的指针为栈区指针;heap标识的指针为堆区指针;如果指针定义前不添加指针区域类型标识符,则默认指针为全局数据区指针。

[0186] 编译器通过增设语法规则,识别出程序中保留的MSU信息,把信息保存在语法树上。供后续步骤使用。

[0187] 编译器进行语法分析时,可通过上述规则分别认定程序中与MSU相关的信息,最终生成语法树、保存MSU信息,其余语法的编译技术与现有技术相同。

[0188] B1-1-2:内存布局及编址方式

[0189] 把属于同一MSU的指令和数据,以页对齐的形式,分别密排链接,指令保存在指令区、数据保存在数据区。所有MSU在同一线性地址空间内,以同一个基址进行统一编址。

[0190] B1-1-3:提取并保存MSU信息:

[0191] 在编译链接阶段,为每个MSU建立以下数据,存储在MSU的数据区:

[0192] 当前MSU的ID;MSU控制对照表;端口匹配表;指向MSU控制对照表的指针变量;指向端口匹配表的指针变量;用以记录MSU栈底地址值的变量。

[0193] 所述当前MSU的ID,保存当前MSU正在运行的MSU的ID值,用以在 MSU控制对照表中找到当前正在运行的MSU的信息。

[0194] 所述MSU控制对照表的信息包括:所有MSU的信息,具体包括:MSU 的ID号、MSU的边界信息、属性信息、端口信息、生效/失效信息。优选的,还包括MSU所属用户类型信息,MSU所属用户标识信息。表中:

[0195] 所述MSU的ID号,通过语法树中保存的不同MSU名生成;

[0196] 所述MSU边界信息,包括:指令区边界信息、全局数据区边界信息、堆区边界信息。对于指令区边界信息、全局数据区边界信息,可以通过统计编译生成的指令和全局数据占用空间大小来确定。对于堆区边界信息,由于编译时无法确定需要建立的堆区大小,可以在对照表中先预留表项,等到运行时需要堆区的时候再临时添加信息;

[0197] 所述MSU属性信息,可以根据语法树中记录的MSU类型信息来设定;

[0198] 所述MSU端口信息包括:MSU的出口信息和MSU的入口信;

[0199] 所述MSU的出口信息,包括:其所属的MSU的ID、出口号、出口地址值;其中出口号为每一个出口的唯一编号,出口地址值为MSU间调用/返回指令所在地址值;

[0200] 所述MSU的入口信息,包括:其所属的MSU的ID、入口号、入口地址值;其中入口号为每一个入口的唯一编号,入口地址值为MSU间调用指令的下一条指令地址值,以及端口函数的第一条指令的地址值;

[0201] 所述生效/失效信息,通过语法树节点中记录的生效/失效标记设置。

[0202] 所述端口匹配表,为本MSU调用其它MSU的调用关系集合。其中一个表项,包括:一对有MSU间调用关系的出口和入口。

[0203] 所述指向MSU控制对照表的指针变量,用于在检查指令中,访问MSU控制对照表。

[0204] 所述指向端口匹配表的指针变量,用于在检查指令中,访问端口匹配表。

[0205] 所述用以记录MSU栈底地址值的变量,用于在检查指令中,控制当前MSU 的栈区访问边界。此变量的初始值为对应特权级的栈的栈底地址值。

[0206] 在每个MSU数据区的线性地址空间中,以页对齐的方式预留一段空间,空间大小为页大小的整数倍,将控制对照表设置于其中,其它数据不可存入其中,并保存到可执行文件内。

[0207] B1-2 限定MSU语法访问规则:

[0208] 编译器分析语法树中记载的信息,对不符合MSU访问规则的代码不予生成可执行程序,如符合,进入后续的生成汇编代码、链接的流程。

[0209] B1-3 生成与MSU访问相关的指令:

[0210] 生成的MSU间调用访问转移指令为:call目标地址值。MSU间调用时,不允许通过call指令进行间接转移。

[0211] 生成的MSU间返回访问转移指令为:ret。

[0212] 访问本MSU全局数据、堆数据的指令与访问栈数据的指令一致。

[0213] B2 运行时阶段对MSU信息的处理

[0214] 创建进程时,为每个MSU申请独立的页面,用以加载上述用于边界访问控制的数据,根据进程的用户ID、用户角色类型,设置MSU属性中的MSU所属用户标识信息、MSU所属用户类型信息,页面中不能存在其它内容,为了保证数据的安全,一种优选的方案是:加载后将页面设置为只读,在需要修改这些数据时,关闭只读,修改完成后,再重新设置为只读。

[0215] 创建进程时,由操作系统为进程分配栈区域,一种优选的方案是:栈的大小被设置为实际适用的大小,而非整个线性地址空间的大小,代表栈的共享数据MSU的边界设置为与栈的边界相同。

[0216] 如果操作系统加载程序时,MSU的内存分配布局,与编译链接时,确定的用于边界访问控制的数据不同,则需将该数据改为与实际相符。

[0217] MSU中程序执行时,如果需要申请/释放堆空间,则通过专用系统调用进入内核,由内核中专用程序为其申请/释放堆空间,并相应修改MSU控制对照表中堆区域边界值。

[0218] MSU中程序执行时,如果需要添加/删除MSU,则通过专用系统调用进入内核,由内核中专用程序为其添加/删除MSU,并修改相应用于边界访问控制的数据。

[0219] 以上所述仅为本发明的较佳实施例而已,并非用于限定本发明的保护范围。凡在本发明的精神和原则之内所作的任何修改、等同替换、改进等,均包含在本发明的保护范围内。

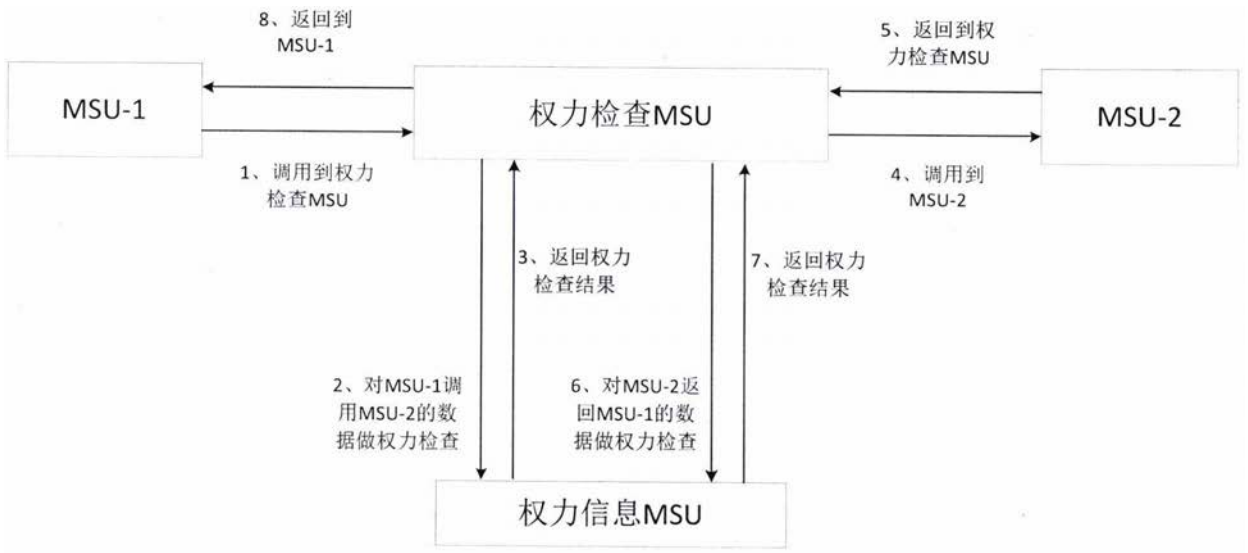

8、返回到
MSU-1
5、返回到权
力检查MSU
MSU-1
权力检查MSU
MSU-2
1、调用到权力
检查MSU
4、调用到
MSU-2
3、返回权力
检查结果
7、返回权力
检查结果
2、对MSU-1调
用MSU-2的数
据做权力检查
6、对MSU-2返
回MSU-1的数
据做权力检查
权力信息MSU

图1