(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 111448552 B
(45)授权公告日 2023.12.01

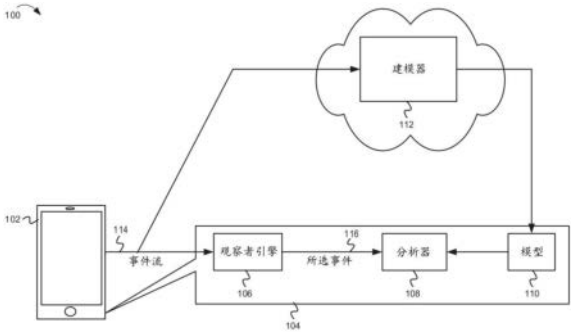
(21)申请号 201880063841.3
(22)申请日 2018.09.29
(65)同一申请的已公布的文献号
申请公布号 CN 111448552 A
(43)申请公布日 2020.07.24
(30)优先权数据
62/565,917 2017.09.29 US
16/141,268 2018.09.25 US
(85)PCT国际申请进入国家阶段日
2020.03.30
(86)PCT国际申请的申请数据
PCT/US2018/053656 2018.09.29
(87)PCT国际申请的公布数据
WO2019/068054 EN 2019.04.04
(73)专利权人 爱维士软件有限责任公司
地址 捷克共和国布拉格
(72)发明人 H.卢 F.哈夫利切克 P.索尔 T.波普
(74)专利代理机构 北京市柳沈律师事务所 11105
专利代理师 张晓明
(51)Int.Cl.
G06F 11/30(2006.01)
H04W 12/12(2006.01)
G06F 21/56(2006.01)
(56)对比文件
US 2007118559 A1,2007.05.24
CN 103559445 A,2014.02.05
US 6205482 B1,2001.03.20
US 2013167231 A1,2013.06.27
CN 106663172 A,2017.05.10
US 2014316984 A1,2014.10.23
US 8739283 B1,2014.05.27
CN 106716382 A,2017.05.24
US 2017222979 A1,2017.08.03
US 2016371152 A1,2016.12.22
US 2003041124 A1,2003.02.27
WO 2012075336 A1,2012.06.07
审查员 吴峰
权利要求书3页 说明书11页 附图7页

(54)发明名称

设备事件的观察和分类

(57)摘要

系统和方法观察和分类设备事件。可以基于机器学习和训练方法确定包含要观察的特征组的模型。客户端应用可以向操作系统服务发出事务请求。可以确定当前是否正在观察操作系统服务、与事务请求相关联的方法以及客户端应用。响应于确定正在观察操作系统服务、与事务请求相关联的方法以及客户端应用，可以修改与客户端应用相关联的行为向量，以指示该方法表示的特征与客户端应用相关联。行为向量可用于确定客户端应用程序是否为恶意软件。



CN 111448552 B

1.一种用于观察设备事件的方法,所述方法包括:

从客户端应用接收对操作系统服务的事务请求;

通过确定所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用是否正在被工具化以及所述客户端应用是否在被评估来确定是否正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用;

响应于确定正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用,修改与所述客户端应用相关联的行为向量并处理所述事务请求;以及

响应于确定没有正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用中的至少一个,在不修改与所述客户端应用相关联的所述行为向量的情况下处理所述事务请求。

2.根据权利要求1所述的方法,还包括:

接收一组要观察的特征,所述一组要观察的特征与一个或多个服务的一个或多个方法相关联;

其中,确定是否正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用包括:确定与所述事务请求相关联的所述方法是否包括所述一个或多个服务的一个或多个方法中的一个。

3.根据权利要求2所述的方法,其中,所述事务请求与所述一组要观察的特征中的特征相关联,并且其中,修改与所述客户端应用相关联的所述行为向量包括:在所述行为向量中设置值以指示与所述事务请求相关联的所述方法已被调用。

4.根据权利要求1所述的方法,还包括至少部分地基于与所述客户端应用相关联的所述行为向量来确定所述客户端应用包括恶意软件。

5.根据权利要求4所述的方法,其中,至少部分地基于与所述客户端应用相关联的所述行为向量来确定所述客户端应用包括恶意软件包括:将与所述客户端应用相关联的所述行为向量与先前观察到的与所述恶意软件相关联的行为向量关联。

6.根据权利要求1所述的方法,其中,所述操作系统服务和与所述事务请求相关联的所述方法中的至少一个在Android中被实现为绑定器的组件。

7.根据权利要求1所述的方法,其中,所述操作系统服务和与所述事务请求相关联的所述方法中的至少一个在iOS中被实现为XPC的组件。

8.一种非暂时性机器可读存储介质,其上存储有用于观察设备事件的计算机可执行指令,所述计算机可执行指令使一个或多个处理器执行操作,包括:

从客户端应用接收对操作系统服务的事务请求;

通过确定所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用是否正在被工具化以及所述客户端应用是否在被评估来确定是否正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用;

响应于确定正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用,修改与所述客户端应用相关联的行为向量并处理所述事务请求;以及

响应于确定没有正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用中的至少一个,在不修改与所述客户端应用相关联的所述行为向量的情况下处理所述事务请求。

9.根据权利要求8所述的非暂时性机器可读存储介质,其中,所述计算机可执行指令还包括指令,用以:

接收一组要观察的特征,所述一组要观察的特征与一个或多个服务的一个或多个方法相关联;

其中,用以确定是否正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用的所述计算机可执行指令包括指令,用以:确定与所述事务请求相关联的所述方法是否包括所述一个或多个服务的一个或多个方法中的一个。

10.根据权利要求9所述的非暂时性机器可读存储介质,其中,所述事务请求与所述一组要观察的特征中的特征相关联,并且其中,修改与所述客户端应用相关联的所述行为向量包括:在所述行为向量中设置值以指示与所述事务请求相关联的所述方法已被调用。

11.根据权利要求8所述的非暂时性机器可读存储介质,其中,所述计算机可执行指令还包括指令,用以:至少部分地基于与所述客户端应用相关联的所述行为向量来确定所述客户端应用包括恶意软件。

12.根据权利要求11所述的非暂时性机器可读存储介质,其中,用以至少部分地基于与所述客户端应用相关联的所述行为向量来确定所述客户端应用包括恶意软件的计算机可执行指令包括指令,用以:将与所述客户端应用相关联的所述行为向量与先前观察到的与所述恶意软件相关联的行为向量关联。

13.根据权利要求8所述的非暂时性机器可读存储介质,其中,所述操作系统服务和与所述事务请求相关联的所述方法中的至少一个在Android中被实现为绑定器的组件。

14.一种用于观察设备事件的系统,所述系统包括:

一个或多个处理器;以及

一种非暂时性机器可读介质,其上存储了计算机可执行指令以使一个或多个处理器执行以下操作:

从客户端应用接收对操作系统服务的事务请求;

通过确定所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用是否正在被工具化以及所述客户端应用是否在被评估来确定是否正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用;

响应于确定正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用,修改与所述客户端应用相关联的行为向量并处理所述事务请求;以及

响应于确定没有正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用中的至少一个,在不修改与所述客户端应用相关联的所述行为向量的情况下处理所述事务请求。

15.根据权利要求14所述的系统,其中,所述计算机可执行指令还包括指令,用以:

接收一组要观察的特征,所述一组要观察的特征与一个或多个服务的一个或多个方法相关联;

其中,用以确定是否正在观察所述操作系统服务、与所述事务请求相关联的方法以及所述客户端应用的所述计算机可执行指令包括指令,用以:确定与所述事务请求相关联的所述方法是否包括所述一个或多个服务的一个或多个方法中的一个。

16.根据权利要求15所述的系统,其中,所述事务请求与所述一组要观察的特征中的特

征相关联，并且其中，修改与所述客户端应用相关联的所述行为向量包括：在所述行为向量中设置值以指示与所述事务请求相关联的所述方法已被调用。

17.根据权利要求14所述的系统，其中，所述计算机可执行指令还包括指令，用以：至少部分地基于与所述客户端应用相关联的所述行为向量来确定所述客户端应用包括恶意软件。

18.根据权利要求17所述的系统，其中，用以至少部分地基于与所述客户端应用相关联的所述行为向量来确定所述客户端应用包括恶意软件的计算机可执行指令包括指令，用以：将与所述客户端应用相关联的所述行为向量与先前观察到的与所述恶意软件相关联的行为向量关联。

19.根据权利要求14所述的系统，其中，所述操作系统服务和与所述事务请求相关联的所述方法中的至少一个在Android中被实现为绑定器的组件。

20.根据权利要求14所述的系统，其中，所述操作系统服务和与所述事务请求相关联的所述方法中的至少一个在iOS中被实现为XPC的组件。

设备事件的观察和分类

技术领域

[0001] 本公开通常涉及用于设备安全的系统和方法，并且更具体地，涉及观察和分类设备事件。

背景技术

[0002] 随着时间的流逝，智能电话变得越来越强大，并且其使用量也在增加。但是，由于用途的增加，智能手机已成为恶意软件的更有吸引力的目标。Malware是“恶意软件(malicious software)”的缩写，是可用于在用户不知情或未经用户同意的情况下破坏设备操作、毁坏数据、收集敏感信息或访问私人计算机系统的软件。此类恶意软件的示例包括软件病毒、特洛伊木马、根程序病毒包(rootkits)、勒索软件等。正确识别哪些文件包含恶意软件以及哪些文件是良性的可能是一项艰巨的任务，因为恶意软件开发人员经常混淆恶意软件的各种属性以试图避免被反恶意软件软件检测到。

附图说明

[0003] 为了更好地理解本发明的主题，可以参考附图，其中：

[0004] 图1是示出根据实施例的用于观察和分类设备事件的系统的组件之间的数据流的框图。

[0005] 图2是示出根据实施例的用于观察和分类设备事件的软件环境的框图。

[0006] 图3是示出根据实施例的在用于观察和分类设备事件的系统中的应用与服务之间的事件流的框图。

[0007] 图4是示出根据实施例的用于在观察和分类设备事件的系统中更新动态模型的软件环境的组件的框图。

[0008] 图5是示出根据实施例的用于初始化设备事件的观察和分类的方法的流程图。

[0009] 图6是示出根据实施例的用于过滤观察到的和分类的设备事件的方法的流程图。

[0010] 图7是可以在其上执行本发明主题的实施例的计算机系统的示例实施例的框图。

具体实施方式

[0011] 在示例实施例的以下详细描述中，参考附图，该附图形成本发明的示例实施例的一部分，并且其中通过图示的方式示出了可以实践本发明主题的特定示例实施例。对这些实施例进行足够详细的描述，以使本领域技术人员能够实践本发明的主题，并且应该理解，可以利用其他实施例，并且可以在不脱离本发明主题的范围的情况下进行逻辑、机械、电气和其他改变。

[0012] 以下详细描述的某些部分是根据对计算机存储器内的数据位的操作的算法和符号表示来呈现的。这些算法描述和表示是数据处理领域的技术人员用来最有效地将他们工作的实质传达给本领域其他技术人员的方式。这里的算法通常被认为是导致期望结果的自洽序列。这些步骤是需要对物理量进行物理操纵的步骤。通常，尽管不是必须的，这些量采

用能够被存储、传输、组合、比较和以其他方式操纵的电信号或磁信号的形式。主要出于通用的原因，有时已经证明将这些信号称为比特、值、元素、符号、字符、术语、数字等是方便的。然而，应该记住，所有这些和类似术语都与适当的物理量相关联，并且仅仅是应用于这些量的方便标签。除非另有明确说明，否则从以下讨论中可明显看出，诸如“处理”或“计算”或“运算”或“确定”或“显示”等术语是指计算机系统或类似计算设备的动作和过程，其将表示为计算机系统的寄存器和存储器内的物理(例如，电子)量的数据操纵和变换为类似地表示为计算机系统存储器或寄存器或其他此类信息存储、传输或显示设备内的物理量的其他数据。

[0013] 各种实施例的描述仅被解释为示例，并未描述本发明主题的每个可能实例。使用当前或未来技术的组合可以实现许多替换方案，这仍然属于权利要求的范围。因此，以下详细描述不应被视为具有限制意义，并且本发明主题的范围仅由所附权利要求限定。

[0014] 实际上，在诸如智能电话等移动设备上使用各种技术来检测恶意软件。一种这样的技术，即行为分析，使用对进程行为的观察来启发式检测恶意意图。实际上，恶意软件检测系统的设计应在感兴趣的特定位置探测系统，并将其行为与已知恶意软件的行为进行比较。恶意软件检测系统分两个步骤执行检测。在第一步期间，执行观察以收集感兴趣的事件。在第二步中，将感兴趣的事件聚合起来进行过滤和分类。行为向量是由聚合事件的分类产生的，可以将其进一步处理为启发式或置信度水平。可以将置信水平与客观指标进行比较，以确定观察是否为恶意行为。可以将观察到的恶意行为与目标应用相关联，以确定应用是恶意的还是良性的、恶意的或不确定的。

[0015] 定义

[0016] 如本文所用，挂钩(hook)被定义为一小段代码，用于收集有关系统内“特征”使用情况的信息。挂钩在程序执行期间获得的信息可以提供有关特征的数据。当在机器处理器上执行时，挂钩可以将程序执行重定向到为执行分析而编写的一组指令。

[0017] 如本文所使用的，行为特征被定义为将在程序运行期间生成感兴趣事件的模块或代码行。通过在感兴趣位置的感兴趣系统中添加一个或多个“挂钩”，观察者将根据需要生成特征事件的聚合，以生成行为向量。

[0018] 如本文所使用，行为向量被定义为定义感兴趣行为的一组行为特征。机器学习模型可以使用任何线性或非线性方法来转换行为向量以产生结果。结果用作恶意软件检测的启发。

[0019] 图1是示出根据实施例的用于观察和分类设备事件的系统100的组件之间的数据流的框图。在一些实施例中，系统100包括具有分析软件环境104的计算设备102，分析软件环境104包括观察者引擎106、分析器108和模型110。系统100还可以包括建模器112，其可以基于计算设备102，或者可以在计算设备102外部，如图1所示。

[0020] 计算设备102可以是任何类型的智能电话。然而，本发明的主题不限于智能电话，并且在替代实施例中，计算设备102可以是平板计算机、媒体播放器(例如，MP3播放器)、膝上型计算机、台式计算机等。在一些实施例中，计算设备102执行Android操作系统。在替代实施例中，计算设备102执行Linux操作系统的版本。在另外的替代实施例中，计算设备可以执行iOS操作系统的版本。

[0021] 观察器引擎106可以获取和处理在计算设备102上运行的应用或服务的特征使用

情况。观察器引擎106可以维护每个应用的行为向量，并且可以在检测到应用的特征使用情况时更新行为向量。

[0022] 模型110可以包括定义感兴趣特征的数据。在一些实施例中，模型110可以是包括评估模型和感兴趣的特征的列表的二进制斑点。评估模型可以是接收每个应用的行为向量作为输入的程序，并返回置信度值。在一些实施例中，行为特征的列表描述了系统服务中的工具点、特征的数据类型以及行为向量内部的特征位置。

[0023] 分析器108可以是可以接收模型110、评估模型110并将评估结果通知感兴趣的各方的服务。在一些实施例中，分析器108可以接收模型110、解包从更高级别的服务接收的二进制斑点并将模型分成评估程序和感兴趣的特征列表两部分。评估模型意味着针对每个应用的行为向量运行评估程序以确定置信度值。在一些实施例中，通知感兴趣的各方可以包括通过使用轮询/读取机制或广播或回调方法来提供接口，以警告移动应用已经发生事件，即应用是恶意软件。

[0024] 建模器112可以包括用于生成模型110的机器学习技术、运算和计算方法。建模器112是用于机器学习过程的抽象(abstraction)，该机器学习过程用于确定要使用哪个特征集(成千上万个特征)以及要在评估模型中编程的计算方法(例如，逻辑回归、随机森林)。可以在模拟器场上运行各种过程来训练模型。作为监督学习过程的一部分，建模者可以从这些过程中接收事件数据，并使用该数据来训练模型。

[0025] 在操作期间，在计算设备102上执行的应用和服务可以生成许多事件。可以捕获这些事件并将其放入提供给观察者引擎106和建模器112的事件流114中。在一些实施例中，这些事件包括在应用和服务之间发生的进程间通信事件。观察者引擎106可以将所选事件116提供给分析器108。可以基于感兴趣的特征来选择事件，并且可以动态地更新这些感兴趣的特征。

[0026] 下面相对于图2提供关于上述系统的各种组件的更多细节。

[0027] 图2是示出根据实施例的用于观察和分类设备事件的软件环境200的框图。在一些实施例中，软件环境200在Android操作系统环境中运行。该体系结构在用户空间内存中沙箱(sandbox)客户端应用和系统应用。可以通过IPC(进程间通信)、消息传递、共享内存或其他通信设施来促进应用和服务之间的通信。

[0028] 在一些实施例中，客户端应用210和212使用经由被称为“绑定器(binder)”的IPC机制发送的消息来与服务(例如，本地服务230)通信。在Android和其他基于Linux的操作系统上提供绑定器IPC机制。绑定器可以包括驻留在内核(kernel)空间中的绑定器驱动器218和驻留在用户空间中的绑定器框架216。绑定器框架216可以被视为在用户空间中运行的“胶水”代码，以促进应用(210和/或212)和服务230与内核驱动器的通信。绑定器一般同时指内核驱动器实现(例如，绑定器驱动器218)和用户空间绑定器框架216。

[0029] 绑定器可以深入了解Android设备在不同离散程度的粗糙度下的运行期间行为。绑定器事务(binder transaction)可以提供两个Android进程之间的通信链的信息密集视图。对于每笔事务，通信协议都包含有关发送方和接收方的元数据，以及来自称为包裹(parcel)的对象的扁平化数据。因此，包裹是存储作为事务对象的数据的容器。对绑定器事务执行的观察可以提供对进程间交互的高度可见性，因为系统调用通常是通过作为绑定器设计一部分的客户端服务器消息传递机制进行的。对于特定的应用(210和/或212)，可以使

用聚合该应用的绑定器调用来分析该应用并确定其行为。

[0030] 在一些实施例中,记录器220、分析器108和观察器引擎106操作以观察和分析绑定器调用以创建行为向量228,该行为向量可用于分析应用的行为并确定该应用是否是恶意的(即,包含恶意软件)。每个行为向量都代表观察到的应用特定于模型的特征。行为向量充当所观察的事件的存储位置。在一些实施例中,每个行为向量可以是计数器的列表,其中计数器可以与模型中的每个特征相关联,并且如果观察到涉及特征的事件,则计数器递增。换句话说,如果为特定服务调用了该方法,并且该方法被认为是"有趣的",意思是说该方法是模型中包括的特征,则在观察器引擎106正在执行和启用的情况下,该特征的计数器将递增。在替代实施例中,行为向量可以包括除了计数器之外或代替计数器而观察到的事件事务的滑动窗口。

[0031] 记录器220可以是一组"挂钩"(用于收集系统内部特征使用情况的小段代码)。这些挂钩可以根据需要分布在Android OS的各种组件周围,以观察所请求的特征。例如,可以分布挂钩以监视由Android OS提供的Android运行期间232和/或本地服务230的各个方面。在一些实施例中,挂钩对应于API调用,其也可以在绑定器框架216中找到。因此,记录器220不需要驻留在内核空间本身中。在替代实施例中,由于某些潜在观察到的特征可能仅在内核中,所以挂钩可以跨越用户空间和内核。由各种挂钩确定的特征使用情况中的变化可以被传递给观察器引擎106以进行进一步处理。以后可以使用特征使用情况来区分恶意应用和干净应用。

[0032] 观察者引擎106可以获取并处理特征使用情况信息。如上所述,由于记录器220的至少一些挂钩可以潜在地驻留在内核空间中,因此在一些实施例中,观察器引擎106也驻留在内核空间中,以确保将观察到的特征传递给整个系统的性能影响忽略不计。可以处理由记录器220的挂钩报告的所观察到的特征,并且可以由观察者引擎106确定是否对相应应用的行为向量228进行了改变。行为向量228可以存储在预分配的存储器结构中。响应于对行为向量228的任何改变,可以通知安全服务222,并且可以将改变的行为向量传递给安全服务222的分析器108组件。观察器引擎106还可以根据分析器108的请求对所观察到的特征的变化做出反应,这将在下面进一步描述。

[0033] 在一些实施例中,在应用的首次启动时,以预定或可配置的阈值时间量和/或事件量执行应用的观察,以期望生成可以被分析的行为向量。在这样的实施例中,除非满足期望的时间或事件阈值,否则不执行分析并且不对应用进行分类。

[0034] 安全服务222可以是观察者引擎106和软件开发工具包(Software Development Kit,SDK)214之间的"连接"模块。在一些实施例中,安全服务222可以包括作为对应用的行为进行分类的子模块的分析器108。在替代实施例中,分析器108可以与安全服务222分开。安全服务222可以从观察者引擎106接收行为向量,并将它们传递给分析器108。然后,如果需要,可以将来自分析器108的结果传播到SDK 214。安全服务222还可以在分析器108中处理模型110的更新。例如,在一些实施例中,可以经由SDK 214将模型110传递给安全服务222。然后,安全服务222可以验证模型110并在分析器108中更新模型。安全服务222还可以将当前模型110请求的要观察的特征通知给堆栈中的相应模块(即记录器220、观察器引擎106等)。下面提供了有关模型更新的工作流的更多详细信息。

[0035] SDK 214可以是安全应用208(即,防病毒或其他反恶意软件的应用)与系统交互的

前端接口。SDK 214可以包括用于认证应用(例如,应用210和/或212)的应用程序接口(API)。SDK 214还可以包括用于控制整个系统行为并更新模型110的API。SDK 214可以将在分析器108中完成的分类的结果传播到安全应用208,以使得它可以对设备上当前发生的事件做出相应的反应。SDK 214可以二进制形式提供给安全应用开发人员适当的文档。另外,在一些实施例中,用于SDK 214的API可以包括用于启用/禁用对应用的监视、隔离应用和/或禁用应用包的方法。

[0036] 可以在各种工作流中使用上述组件和API。这样的工作流可以包括更新模型110、向量化行为观察、评估包/应用矢量上的模型110以及向安全应用208通知有关该应用的信息。

[0037] 将模型推入/更新到分析器中

[0038] 安全应用208可以通过发出对SDK 214中的方法的调用来发起该流程,以将模型推送或更新到系统。在一些实施例中,这使得模型110经由绑定器IPC被发送到安全服务222。安全服务222可以验证模型110的签名,将其拆包并将其分为以下组成部分:评估模型本身和要观察的行为特征列表。评估模型110可以是接收每个应用行为向量作为输入并返回置信度值的程序。此值稍后作为事件发送到安全应用208(参阅以下"通知安全应用"流程)。行为特征的列表描述了系统服务(例如,本地服务230和运行期间232)的检测点、特征的数据类型及其在行为向量内的位置。在一些实施例中,使用操作码从相应服务的"绑定器接口"(如果自动生成,也称为Android界面定义语言(AIDL)文件)中唯一标识每个特征。这些操作码是在"绑定器接口"的编译之前或期间定义的。操作码和服务对可以唯一地标识特征。

[0039] 一旦安全服务222解包了模型110,安全服务222就可以通知系统服务新模型110可用,在每个服务上运行的线程(例如,更新器线程408,图4)可以通过读取新特征列表,并在需要时应用工具点来响应。无论是模型110的首次初始化还是后来的更新,此机制都相同。在一些实施例中,当推送新模型110时,可以重置每个应用的行为向量。

[0040] 矢量化行为观察

[0041] 当认为对系统服务的方法调用感兴趣时,即该特征在特定于模型的工具列表中时,将处理此工作流。可以执行几项检查,以确定是否需要考虑该行为。可以按照以下顺序执行这些检查:首先排除最常见的情况,然后将行为代码排除在外,以尽可能快地进行IPC调用。这是理想的,因为它可以最小化观察者引擎106施加的开销。

[0042] 在一些实施例中,系统进行检查以确定是否正在工具化(instrument)系统服务。然后,系统可以检查以确认需要对被拦截的特定方法进行计数。系统可以进一步检查以确认调用者是正在积极评估(即尚未被禁用)的应用。在替代实施例中,系统可以扩展每个检查以进一步包括由特征列表234确定的情况。例如,这种情况可能是,例如,如果调用此绑定器方法的应用是当前的前景(Foreground)应用,或者如果以前已经检测到某些其他特征(通常称为二次事件,或更常见的是非线性事件链)。如上所述,这些检查的顺序可以变化,取决于每个单独的检查如何影响监视IPC事务的开销。

[0043] 一旦通过这些检查,系统便可以在模型110中修改每个应用的行为向量。在一些实施例中,系统可以设置分析器108可以用来确定哪些应用需要再次评估的标志。

[0044] 在包矢量上评估模型

[0045] 在一些实施例中,安全服务222可以等待新的行为信息的通知。每当安全服务222

确定应用的行为向量已改变时，它就可以使用此改变的行为向量作为输入参数来评估模型110。模型110可以包括评估被编译为小型可执行程序的模型，以减少评估过程中的任何开销。在一些实施例中，安全服务222可以直接访问行为向量228的映射，以避免需要请求或复制向量的缓冲器。与典型系统相比，这两个设计决策(编译模型和零复制语义)可以使整个模型评估更加有效。

[0046] 通知安全应用

[0047] 当评估模型返回高于可配置阈值的置信度值时，可以创建包括有关所涉及的包的相关信息的事件，并将其发送到安全应用208(例如，使用绑定器IPC调用)。此回调的接收者可以最终确定如何对这种潜在威胁做出反应。可能会创建与同一应用相关的多个事件。当接收到事件时，安全应用208可以执行几个动作中的任何一个。例如，安全应用208可以使用由SDK 214提供的API来隔离应用(app)，如果该应用被认为是误报，则禁用对应用的监视，或者如果置信度不够高，则不执行任何操作并继续收集行为信息。另外，在一些实施例中，安全应用可以将事件数据上载到建模器112，以便改善模型110的训练。

[0048] 应当注意，尽管已经在Android OS和绑定器的上下文中讨论了图2，但是实施例不限于此。例如，iOS中的IPC框架称为XPC。在特征上与绑定器类似，XPC提供了客户端服务远程过程调用机制，该机制在客户端和服务之间中继调用。XPC也类似于绑定器，因为包裹是可以包含展平对象的容器，XPC将对象展平为序列化的属性列表。因此，可以将iOS XPC视为等效于Android中的绑定器的iOS。结果，受益于本公开的本领域普通技术人员将理解，本文描述的发明主题的各方面可以应用于iOS和XPC。

[0049] 此外，应当注意，图2示出了其中观察者引擎106在内核空间中执行并且行为向量被存储在内核空间中的实施例。在这些实施例中，观察者引擎106可以实现为设备，并且UNIX ioctl调用可以用于与观察者引擎106通信。在替代实施例中，观察者引擎106和行为向量228之一或两者都可以驻留在用户空间中作为Android OS的一部分。在这些实施例中，观察者引擎106可以被实现为OS守护进程(daemon)并且可以通过UNIX套接字(socket)或其他IPC机制来访问。

[0050] 图3是示出根据实施例的在用于观察和分类设备事件的系统中的应用302与服务304之间的事件流的框图。应用302可以发起对由服务304提供的服务的请求。可以经由通信1将请求提供给服务事务框架306。在一些实施例中，服务事务框架306可以是Android绑定器框架。在替代实施例中，服务事务框架306可以是iOS XPC。应用302与服务事务框架306之间的接口可以执行数据的编组(marshalling)和序列化，以便通信1中的数据采用服务事务框架306期望的格式。

[0051] 服务事务框架306可以经由通信2将请求转发到内核IPC组件308。在一些实施例中，内核IPC组件308可以是绑定器驱动器，并且可以使用对驱动器的ioctl系统调用来执行通信2。服务事务框架306可以执行任何所需的数据的编组和序列化，以便数据采用内核IPC组件308期望的格式。

[0052] 内核IPC组件308中的分派器(dispatcher)(未显示)经由通信3将请求转发到过滤器310。该请求采用服务事务框架306期望的格式。在某些方面，过滤器310可以是绑定器调用IPCThreadState:executeCommand()的修改版本。过滤器310可以接收通信3，并通过测试指向挂钩表的指针来检查挂钩表的存在。如果指针存在，则将挂钩表作为先前事务请求

的一部分加载，并且过滤器可以访问该表。挂钩表包括对应于要在特征列表中指定的要观察的特征的挂钩表。在一些实施例中，过滤器310可以是替代框架的标准绑定器版本的修改后的绑定器框架的一部分。在替代实施例中，过滤器310可以是修改的XPC框架的一部分。除了过滤器310之外，修改的框架还可以包括执行“挂接”和运行分析器108的修改。

[0053] 过滤器310可以检查请求中提供的要进行事务的事务代码是否在挂钩表的边界内。在一些实施例中，事务代码包括由来自称为AIDL的RPC接口的操作码表示的“数字事务代码”。每个操作码唯一地定义来自服务的单个可调用方法。如果在挂钩表中找到该代码，则可以记录事务请求(例如，绑定器或XPC调用)。换句话说，如果设置了操作码索引，则将挂接操作码。

[0054] 因为用于挂接的代码位于请求路径上，所以期望程序尽快恢复其正常操作。因此，在一些实施例中，使用预计算技术来做出消除处理开销的措施。另外，在一些实施例中，在挂接期间不执行存储器分配、搜索或散列。

[0055] 过滤器310经由通信4将请求转发到服务事务框架306。然后，服务事务框架306经由通信5将请求转发到服务304。

[0056] 服务304可以处理该请求，并且请求的结果可以经由通信6-9通过服务事务框架306和内核IPC组件308返回给应用302。

[0057] 将过滤器310放置在内核IPC组件308中的分派器和服务事务框架306之间可能是理想的，因为它可以确保所有事务请求都将通过过滤器310，并且可以观察到模型的特征列表中的所有事务请求。在过滤点，分类过滤器310可以遵循规则，以忽略不在特征列表中的、表示调用者特征的事务请求。

[0058] 图4是示出根据实施例的用于在观察和分类设备事件的系统中更新动态模型的组件的框图。在图4所示的示例中，服务A414和服务B416这两个服务向运行在计算设备上的应用提供服务。服务A414和服务B416可以具有各自的线程池(410、412)，该线程池包括执行其相应服务所提供的功能的可用分派器线程(404、406)的组。

[0059] 在一些实施例中，可以通过在服务(例如，服务A414、服务B416)的运行期间加载特征列表234来修改聚合的行为向量402的组成。在一些方面，聚合的行为向量402包括每个应用的所有行为向量的聚合，其存储在由观察者引擎106初始化的连续存储器位置中。可以从要在设备启动时加载或在运行期间更新的机器学习模型110的二进制部分中存储和更新特征列表234。可以将特征列表234动态地加载到过滤器310(图3)中，并用于观察应用和服务之间的事务(例如，绑定器或XPC事务)。只能对匹配过滤规则的事务请求执行挂接。替代在系统中的客户端和服务器之间的代码路径中放置静态挂钩(即像以前的方法那样在客户端函数调用中)，在服务端的IPC分派器的接收端上选择目标过滤点。在分派之前和从内核IPC组件308(图3)接收到IPC调用之后，过滤器310测试是否挂接该调用。如果过滤器确定调用将被挂接，则该挂钩将在行为向量402中设置标志，以用信号通知已观察到特征。

[0060] 更新器线程408等待标志改变，如果改变，则更新来自观察者引擎106的挂钩表。当将更新的模型110加载到安全性驱动器208中时，可以设置标志，该标志指示线程取消引用旧的挂钩表并引用新的挂钩表。在一些实施例中，使用futex(fast user space mutex，快速用户空间互斥)调用来使更新器线程408从调度循环中移出，直到发生这种改变为止。也就是说，更新器线程408通常不会浪费任何CPU周期。

[0061] 图5是示出根据实施例的用于初始化设备事件的观察和分类的方法的流程图500。在某些方面，该方法可以构成由计算机可执行指令组成的计算机程序。通过参考流程图来描述该方法，使得本领域技术人员可以开发包括这样的指令的程序，以在合适的处理器(执行来自计算机可读介质的指令的计算机的一个或多个处理器)上执行该方法。图5所示的方法包括执行本发明的示例性实施例的操作环境可以采取的动作。

[0062] 在一些实施例中，当接收到针对服务的第一绑定器事务时，图5所示的方法由服务执行。例如，图5的方法可以在绑定器类构造者(constructor)中执行。

[0063] 在框502，服务连接到安全驱动器208。在一些实施例中，当在服务中创建新的绑定器服务对象时，服务连接到安全驱动器。

[0064] 在框504，服务映射特征(即，挂钩)列表。在一些实施例中，服务在初始化时访问绑定器驱动器以映射存储特征列表的共享存储器缓冲区。初始化期间共享存储器缓冲区的存储器映射允许服务在没有任何复制语义的情况下引用特征列表，并且允许服务在服务初始化期间仅创建一次引用。加载新模型时，更新器线程(图4的408)重新创建对特征列表的引用。

[0065] 在框506，服务映射应用行为向量。如上所述，在行为向量中跟踪应用“行为”。在一些实施例中，感兴趣的服务中的分派器线程(图4，404和406)执行行为向量修改。在这样的实施例中，服务的分派器线程(图4，404和406)或更一般地，服务，具有对存储位置的引用，在该存储位置中跟踪所有应用的“行为”(例如，行为向量228)。在一些实施例中，存储位置是由内核初始化并且由服务通过ioctl调用引用的共享存储器映射。在一些实施例中，存储位置可以是每个应用以固定大小分配的连续存储器块。每个应用的偏移可以是应用的UID的倍数。

[0066] 在框508，服务确定针对服务的挂钩。在一些实施例中，挂钩是特定于服务的。特征列表可以是来自与机器学习模型110相关的所有服务的方法调用的列表。每个服务的绑定器接口定义可以包括操作码列表，每个操作码对应于可调用方法。该服务将表(或数组)映射到存储器中，该表(或数组)用于当前服务的操作码通过设置与代表操作码的数组中的位置相对应的一个或多个位而在当前被启用以进行观察。换句话说，为每个要记录的操作码设置标志，为要忽略的每个操作码不设置标志。映射的结果包括挂钩表。

[0067] 服务可以通过将当前服务的可用操作码与特征列表中的服务/操作码列表进行比较，来确定挂钩(或挂钩操作码)。在挂钩表中标记与当前服务匹配的操作码。

[0068] 在框510，服务产生线程(例如，图4的更新器线程408)，该线程的目的是等待标志改变，并且如果改变，则更新来自驱动器的挂钩表。当将更新的模型110加载到安全性驱动器208中时，可以设置标志，该标志指示线程取消引用旧的挂钩表并引用新的挂钩表。如上所述，在一些实施例中，可以使用futex调用来使更新器线程408退出调度循环，直到发生这种改变为止。在一些实施例中，更新程序线程408是针对服务的每个进程而不是每个服务来启动的。

[0069] 图6是示出根据实施例的用于过滤设备事件的方法的流程图600。在一些实施例中，过滤器被设计为在最常见的情况下使开销最小化，并且可以在粒度的三个级别、服务级别、方法级别和应用级别进行测试。

[0070] 在框602，该方法接收应用事件。在一些实施例中，该事件是应用和服务(或服务和

另一服务)之间的绑定器事务。在替代实施例中,事件可以是应用与服务之间的XPC事务。

[0071] 在框604,过滤器310确定服务是否被挂接(即,正在观察服务的特征)。在一些实施例中,这可以在一个指针比较中完成以确定挂钩表/特征列表是否已经被加载。在框604,服务级别过滤器规则检查以查看绑定器调用的接收者是否与加载的模型110中的服务匹配。确定要挂接的服务与要挂接的特征的选择有关。使用建模器112来确定要挂接的特征的选择。如果未挂接服务,则该方法前进到框612以照常处理事务。如果服务被挂接,则该方法进行到框606。

[0072] 在框606,过滤器310确定该方法是否被挂接。在一些实施例中,框606的方法级别过滤器规则检查以查看功能之一是否与所加载的模型110中的功能匹配。在一些实施例中,使用操作码从相应服务的“绑定器接口”(如果自动生成,也称为AIDL文件)中唯一标识特征。这些操作码是在“绑定器接口”的编译之前或期间定义的。由于操作码和服务对可以唯一地标识特征,因此操作码和服务对可以用来确定方法是否已被挂接。如果未挂接该方法,则该方法前进至框612以照常处理事务。如果该方法被挂接,则该方法前进到框608。

[0073] 在框608,过滤器310确定是否正在评估应用。在一些实施例中,在框608处的应用级别过滤器规则检查已知为良性的应用的白名单。白名单可以由绑定器驱动器218维护。可替代地,过滤器规则可以检查要分析的应用的列表。如果没有对该应用进行分析,则该方法前进至框612以照常处理事务。如果正在分析该应用,则该方法前进到框610。

[0074] 在框610,可以将关于事务的事件数据发送到观察者引擎106(图1)。

[0075] 然后,该方法前进到框612以照常处理事务,并且该方法结束。

[0076] 从上面可以理解,处理绑定器调用所需的开销量取决于过滤器处理的深度。对于到达应用级别逻辑的绑定器调用,机器周期的成本相对较高。同样,仅到达服务级别逻辑的绑定器调用在机器周期中具有较低的成本。最好的粒度级别、应用级别可以被认为是开销相对较高的事务,而方法级别和服务级别每个分别需要较少的开销。在图6的流程图中,达到每个过滤器级别的相对频率被加权至服务级别调用。受益于本公开的本领域技术人员将理解,在特定环境中,确定与级别相关联的开销和/或频率使得希望以与图6所示顺序不同的顺序执行检查的情况下,可以调整检查的顺序。

[0077] 发明人已经确定服务级别检查是常见情况,这意味着过滤器逻辑通过服务级别检查的可能性是低概率事件。因此,可以最优化服务级别检查。为了对此进行优化,服务级别逻辑由一次性加载组成,以获取对该服务的挂钩表的引用。任何后续调用仅需要指针比较以确定是否已加载挂钩表。这种便宜的检查可以防止进行更昂贵的第二级和第三级过滤器检查。

[0078] 可以为方法级别过滤器检查推断出相同的开销成本。在一些实施例中,可以在一个加载指令和三个比较中完成框606处的方法级别检查。

[0079] 在框608处的应用级别过滤器检查导致两个较粗粒度检查的正匹配和在白名单中找到应用的负匹配的罕见情况可以被认为是高开销操作。在一些实施例中,可以在大约十八个指令中完成框608处的检查。然而,机器学习模型110可以被优化为使用全部服务的一小子组(subset)和这些服务中所有可能方法的一小子组。通过使用应用的白名单可以进一步优化检查。

[0080] 在一些实施例中,当与可能特征的总数相比时,要观察的特征的总子集可以相对

较小。作为示例而非限制，在特定实施例中观察到的特征的数量可以在大约50,000个可能的特征总数中的150-5000个范围内。受益于本公开的本领域技术人员将理解，观察到的特征的数量和/或可能的特征的数量在不同的实施例中可以变化，并且可以在给定范围之外变化。与特征的总数相比，通过使特征的数量保持相对较小，过滤器逻辑超出应用级别的可能性很小，因为预期用于负匹配情况的决策树概率会很小。由于这个原因，一些实施例的过滤器设计强调在服务级别上的负匹配并且使过滤器检查的深度最小化。这可以通过最小化挂接的特征的数量、最小化挂接的方法的数量以及优化应用白名单来完成。

[0081] 从上面可以理解，一些实施例可以提供计算机和移动设备功能性的改善。例如，与常规系统相比，基于行为的系统可以改善对恶意软件的检测，该常规系统分析代码以寻找与已知恶意软件匹配的代码片段。可以通过不影响恶意软件的主要目的、对代码的微小改动来击败此类常规系统。相反，本文描述的系统和方法可以分析代码的实际行为，而不是代码本身。这可以改善对恶意软件的检测。此外，恶意软件的检测和去除可以导致计算机或移动设备的功能性改善。

[0082] 参考图7，示例实施例扩展到计算机系统700的示例形式的机器，在该机器内可以执行用于使机器执行这里所讨论的任何一种或多种方法的指令。在替换示例实施例中，机器作为独立设备操作或者可以连接(例如，联网)到其他机器。在联网部署中，机器可以在服务器-客户端网络环境中以服务器或客户端机器的能力运行，或者作为端对端(peer-to-peer)(或分布式)网络环境中的对等机器运行。此外，虽然仅示出了单个机器，但术语“机器”还应被视为包括单独或联合执行一组(或多组)指令以执行这里所讨论的任何一种或多种方法的任何机器集合。

[0083] 示例计算机系统700可以包括处理器702(例如，中央处理单元(CPU)、图形处理单元(GPU)或两者)、经由总线708彼此通信的主存储器704和静态存储器706。计算机系统700可以进一步包括触摸屏显示单元710。在示例实施例中，计算机系统700还包括网络接口设备720。

[0084] 永久性存储单元716包括机器可读介质7422，其上存储有一组或多组指令724和由本文所述的任何一个或多个方法或功能实现或使用的数据结构(例如，软件指令)。指令724还可以在由计算机系统700执行期间完全或至少部分地驻留在主存储器704内或处理器702内，主存储器704和处理器702也构成机器可读介质。

[0085] 虽然机器可读介质722在示例实施例中被示为单个介质，但是术语“机器可读介质”可以包括存储一个或多个指令的单个介质或多个介质(例如，集中式或分布式数据库，或相关联的高速缓存和服务器)。术语“机器可读介质”还应被视为包括能够存储、编码或携带由机器执行的指令的任何有形介质，并且使得机器执行本发明的实施例的任何一个或多个方法，或者能够存储、编码或承载由这些指令使用或与之相关联的数据结构。因此，术语“机器可读存储介质”应被视为包括但不限于可以以非暂时方式存储信息的固态存储器和光学与磁性介质，即，能够存储信息的介质。机器可读存储介质的具体示例包括非易失性存储器，包括例如半导体存储器设备(例如，可擦除可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)和闪存设备)；磁盘，诸如内部硬盘和可移动磁盘；磁光盘；和CD-ROM和DVD-ROM磁盘。机器可读存储介质不包括信号。

[0086] 还可以通过通信网络726经由网络接口设备720使用信号传输介质并利用许多众

所周知的传输协议(例如,FTP,HTTP)中的任何一种来发送或接收指令724。通信网络的示例包括局域网(LAN)、广域网(WAN)、因特网、移动电话网络、普通老式电话(POTS)网络和无线数据网络(例如,WiFi和WiMax网络)。术语“机器可读信号介质”应被视为包括能够存储、编码或携带由机器执行的指令的任何暂时无形介质,并且包括数字或模拟通信信号或其他无形介质以便于这样的软件的通信。

[0087] 尽管已经参考具体示例实施例描述了本发明主题的概述,但是在不脱离本发明实施例的更广泛的范围的情况下,可以对这些实施例进行各种修改和改变。这里,本发明主题的这些实施例可以仅仅为了方便而单独地或共同地被称为术语“发明”,并且在实际上公开了多个发明或发明构思的情况下不意图将本申请的范围自愿地限制于任何单个发明或发明构思。

[0088] 从前面的描述显而易见的是,本发明主题的某些方面不受这里所示示例的特定细节的限制,因此本领域技术人员将理解预期的其他修改和应用或其等同物。因此,权利要求旨在覆盖不脱离本发明主题的精神和范围的所有这些修改和应用。因此,显而易见的是,本发明的主题仅由所附权利要求及其等同物限制。

[0089] 提供摘要以符合37 C.F.R.§1.72(b)以允许读者快速确定技术公开的性质和要点。提交摘要时的理解是,它不会用于限制权利要求的范围。

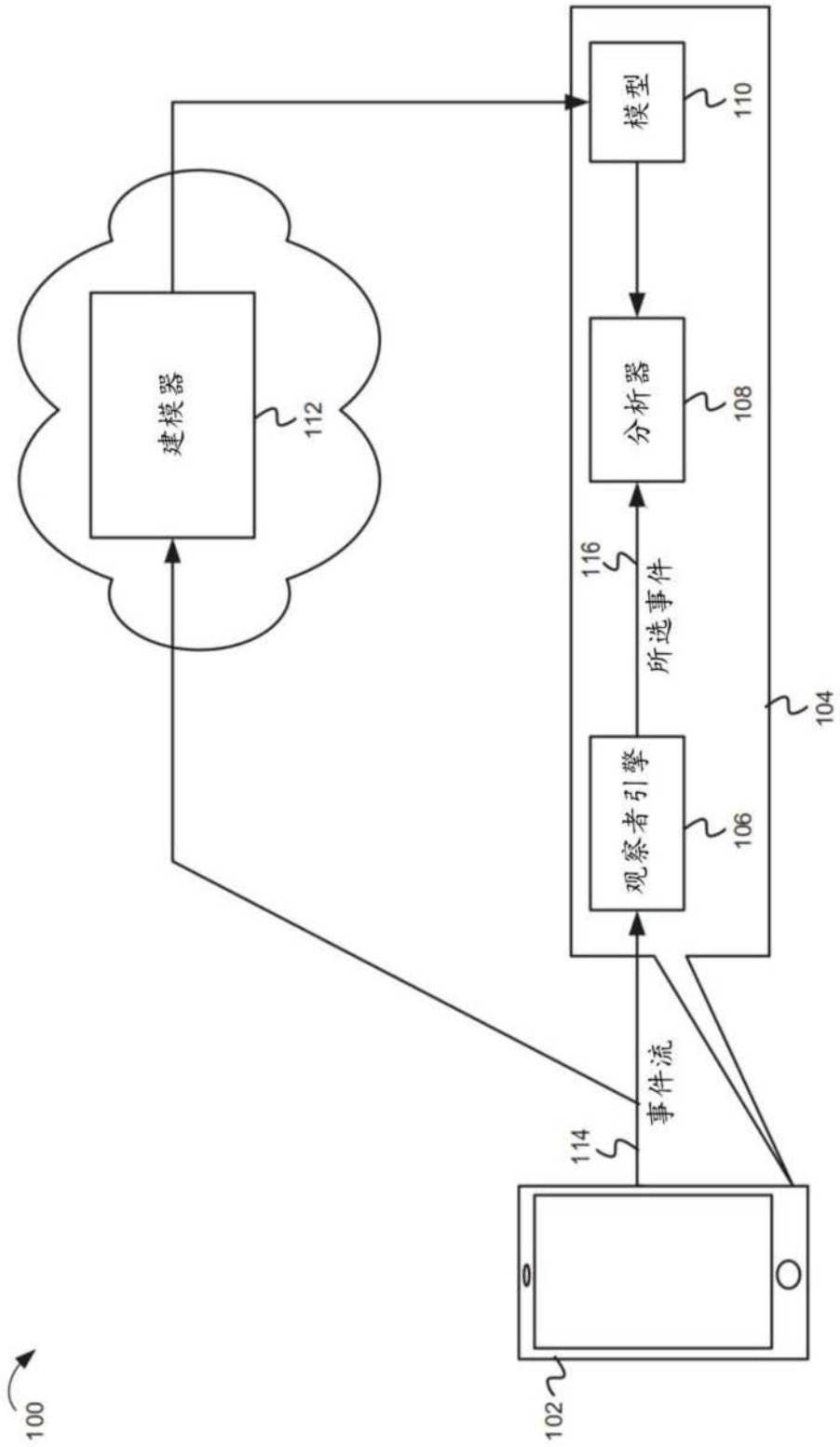


图1

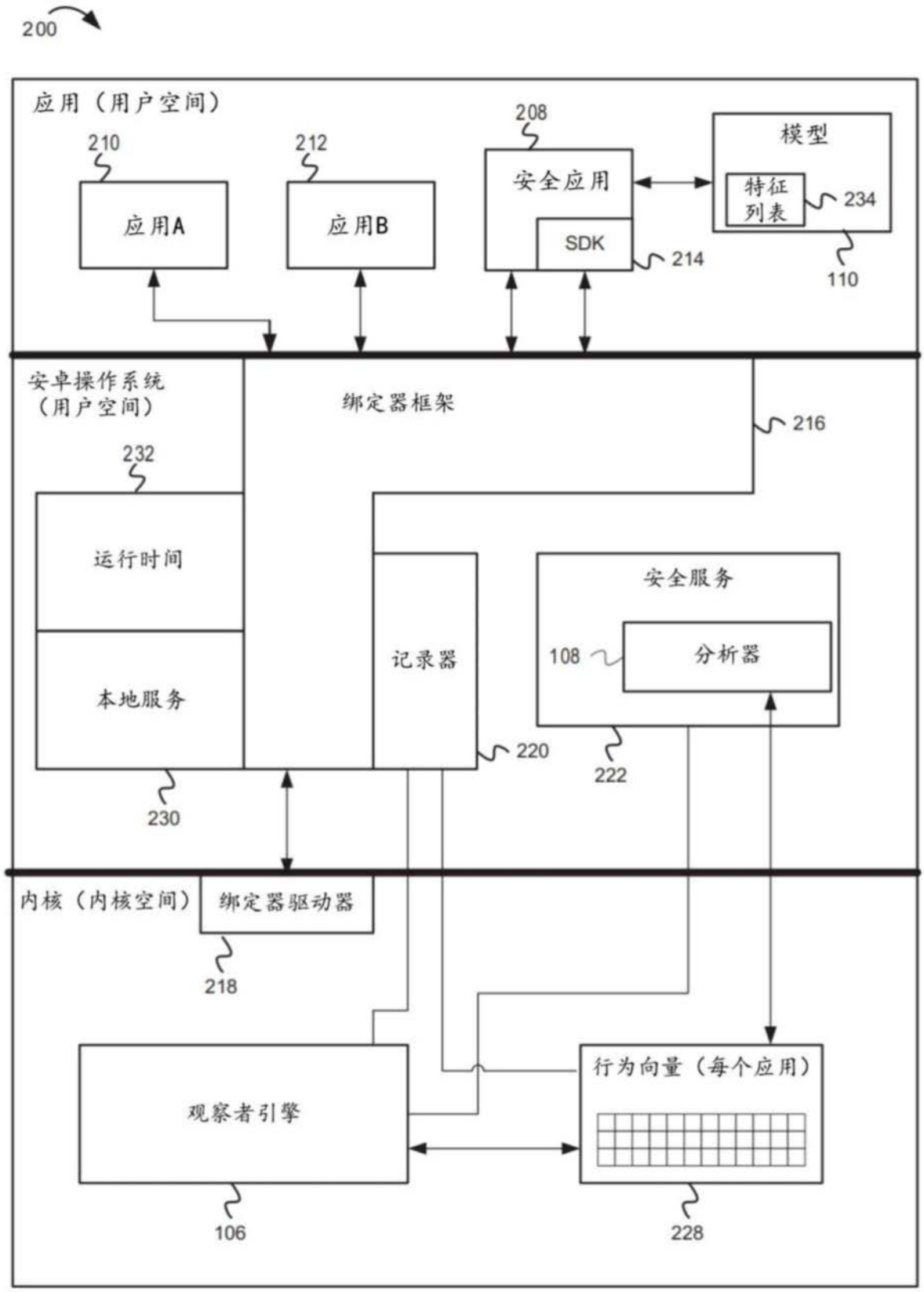

200
应用（用户空间）
210
应用A
212
应用B
208
安全应用
SDK
214
模型
特征列表
234
110
安卓操作系统（用户空间）
绑定器框架
216
232
运行时间
本地服务
230
记录器
220
安全服务
108
分析器
222
内核（内核空间）
绑定器驱动器
218
观察者引擎
106
行为向量（每个应用）
228

图2

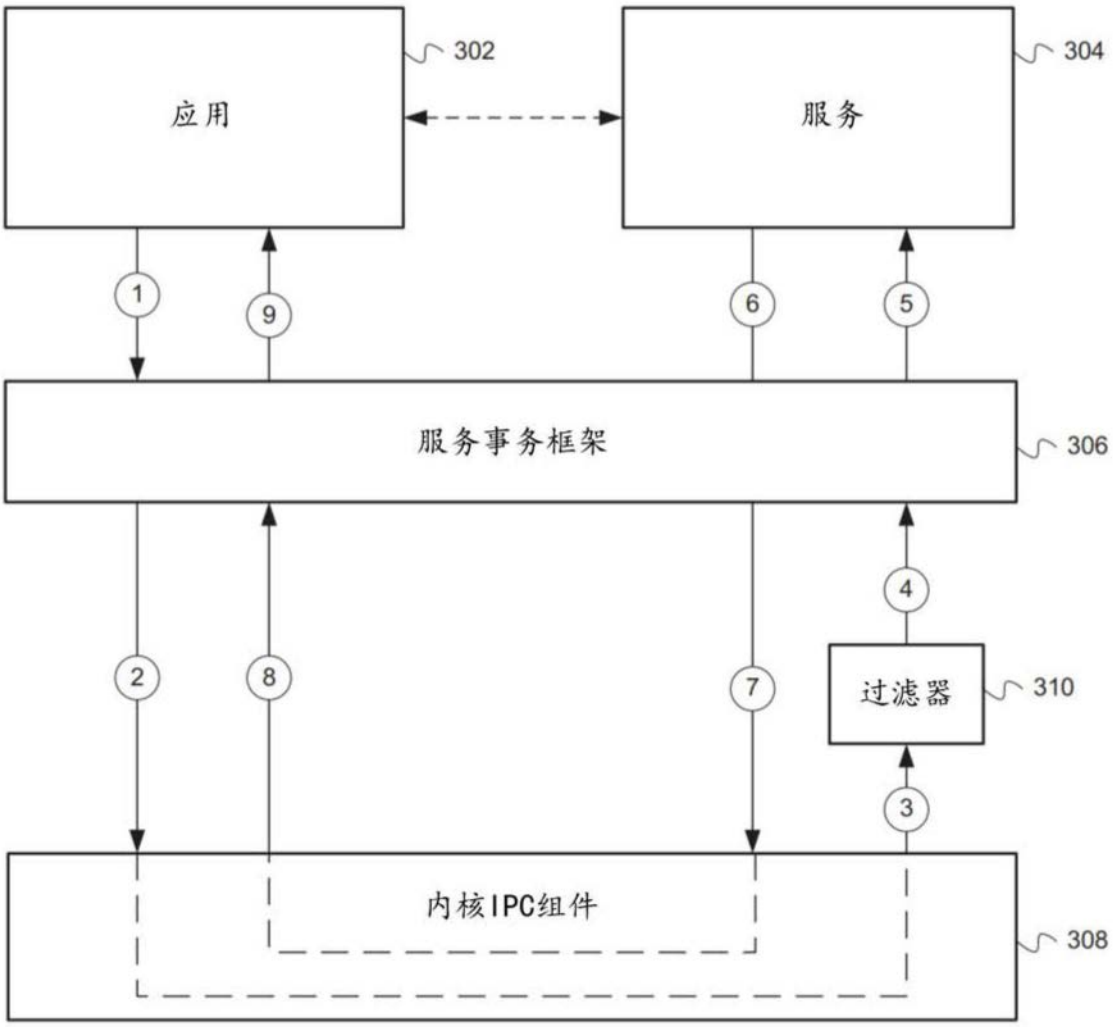


图3

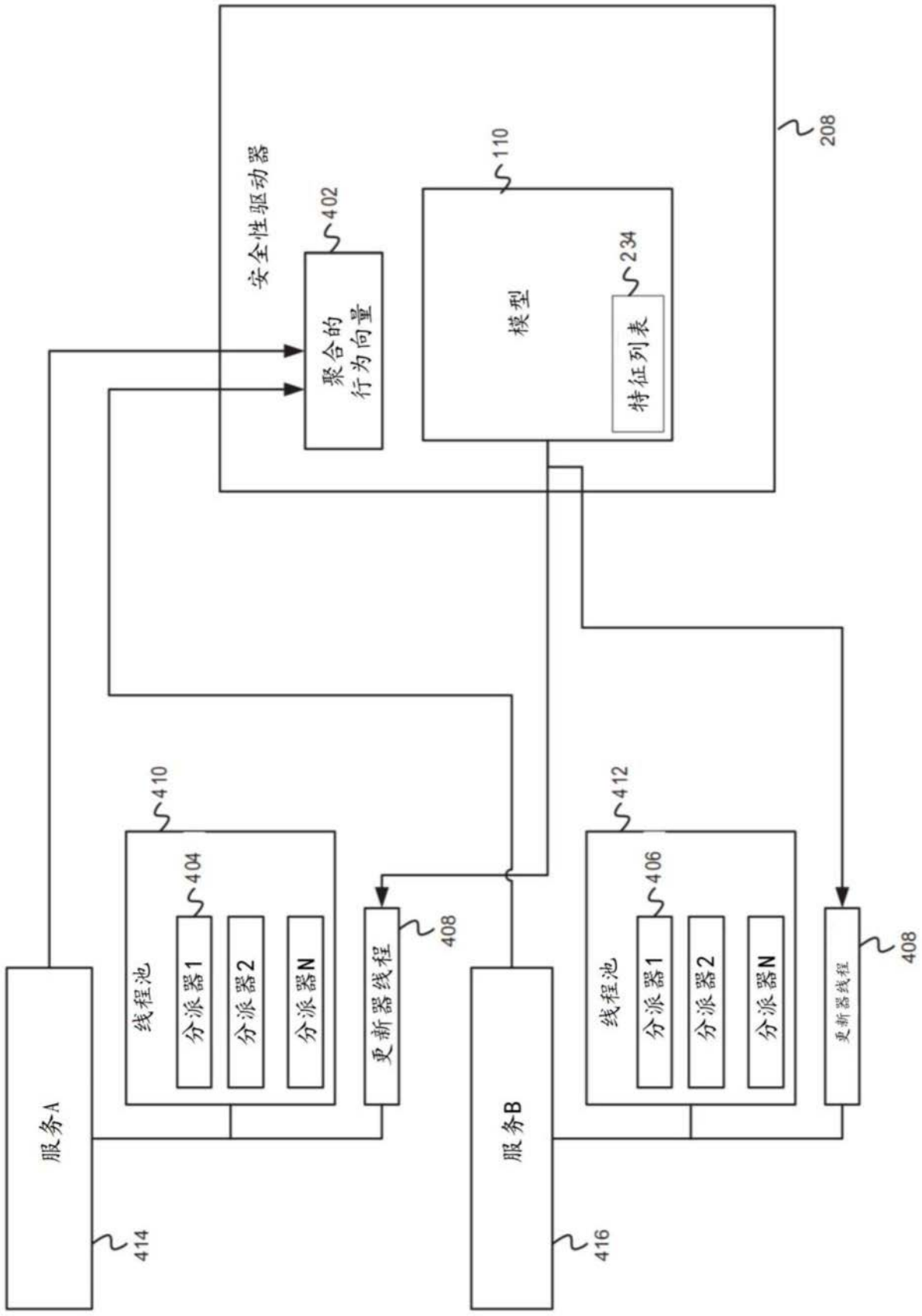

安全性驱动器
208
402
聚合的
行为向量
110
模型
234
特征列表
410
线程池
404
分派器1
分派器2
分派器N
408
更新器线程
412
线程池
406
分派器1
分派器2
分派器N
408
更新器线程
服务A
414
服务B
416

图4

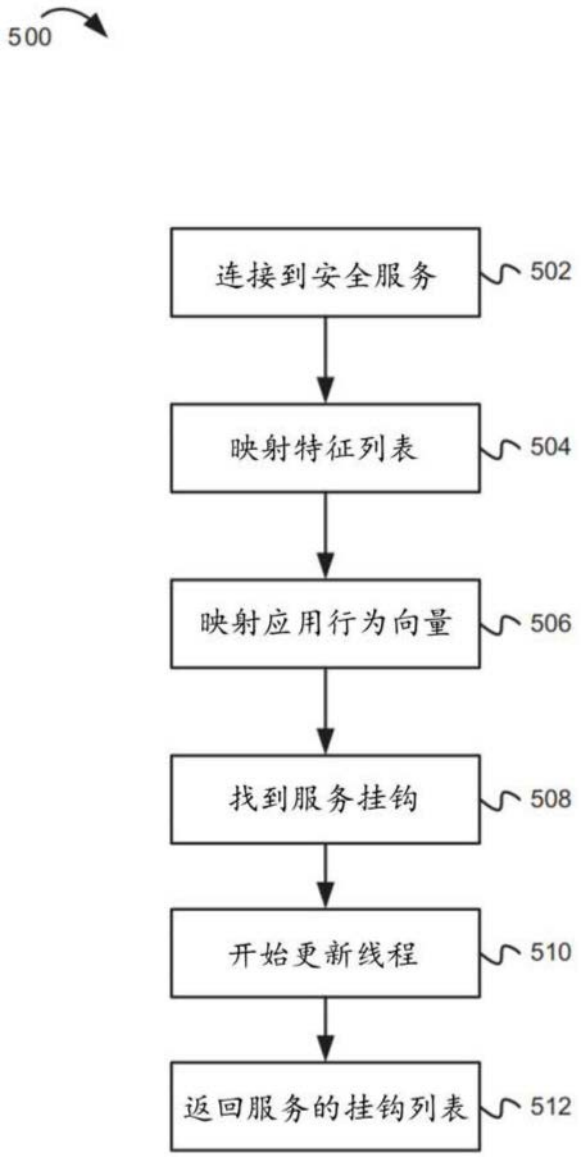

500
连接到安全服务
502
映射特征列表
504
映射应用行为向量
506
找到服务挂钩
508
开始更新线程
510
返回服务的挂钩列表
512

图5

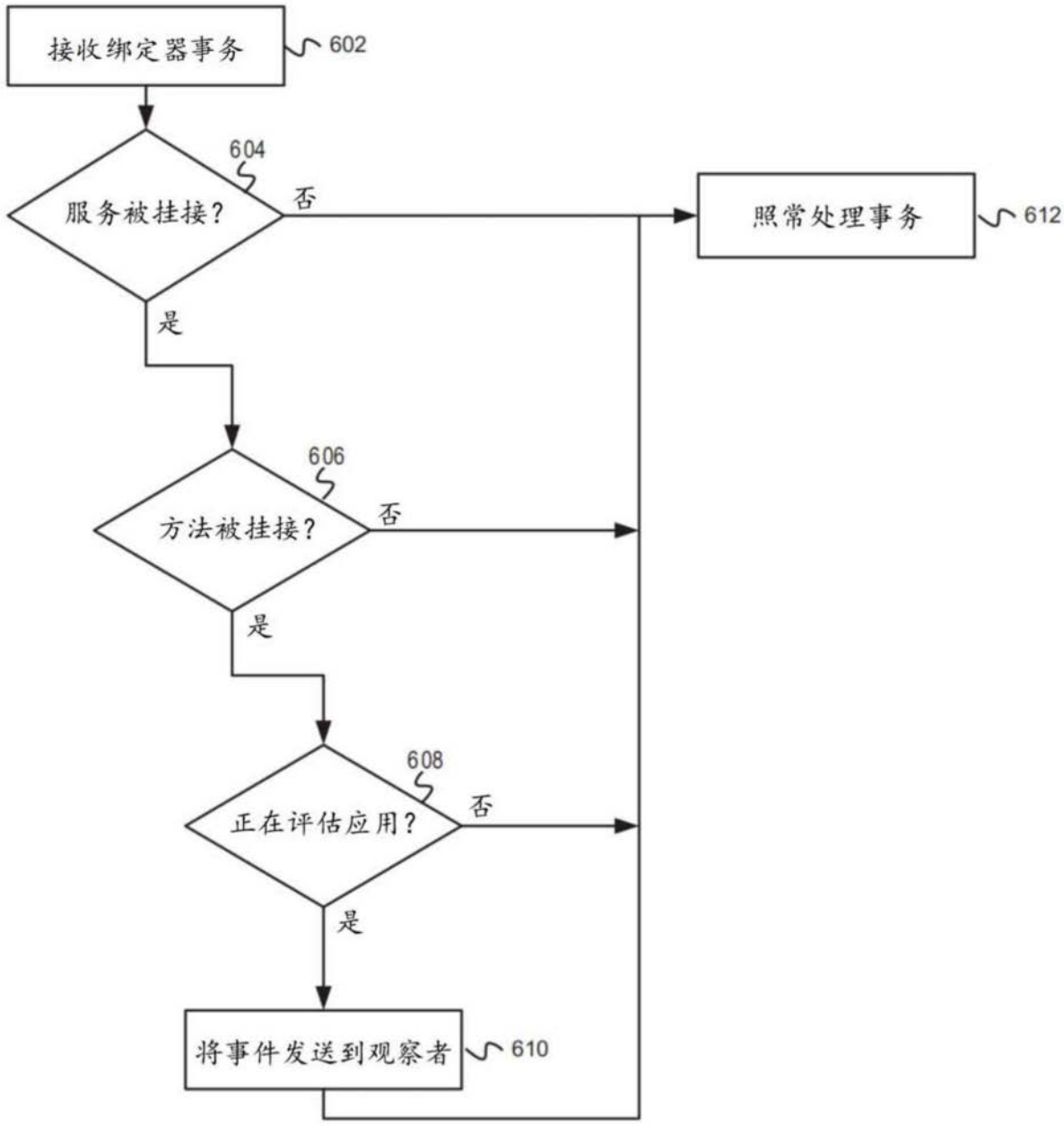

接收绑定器事务
602
604
服务被挂接?
否
照常处理事务
612
是
606
方法被挂接?
否
是
608
正在评估应用?
否
是
将事件发送到观察者
610

图6

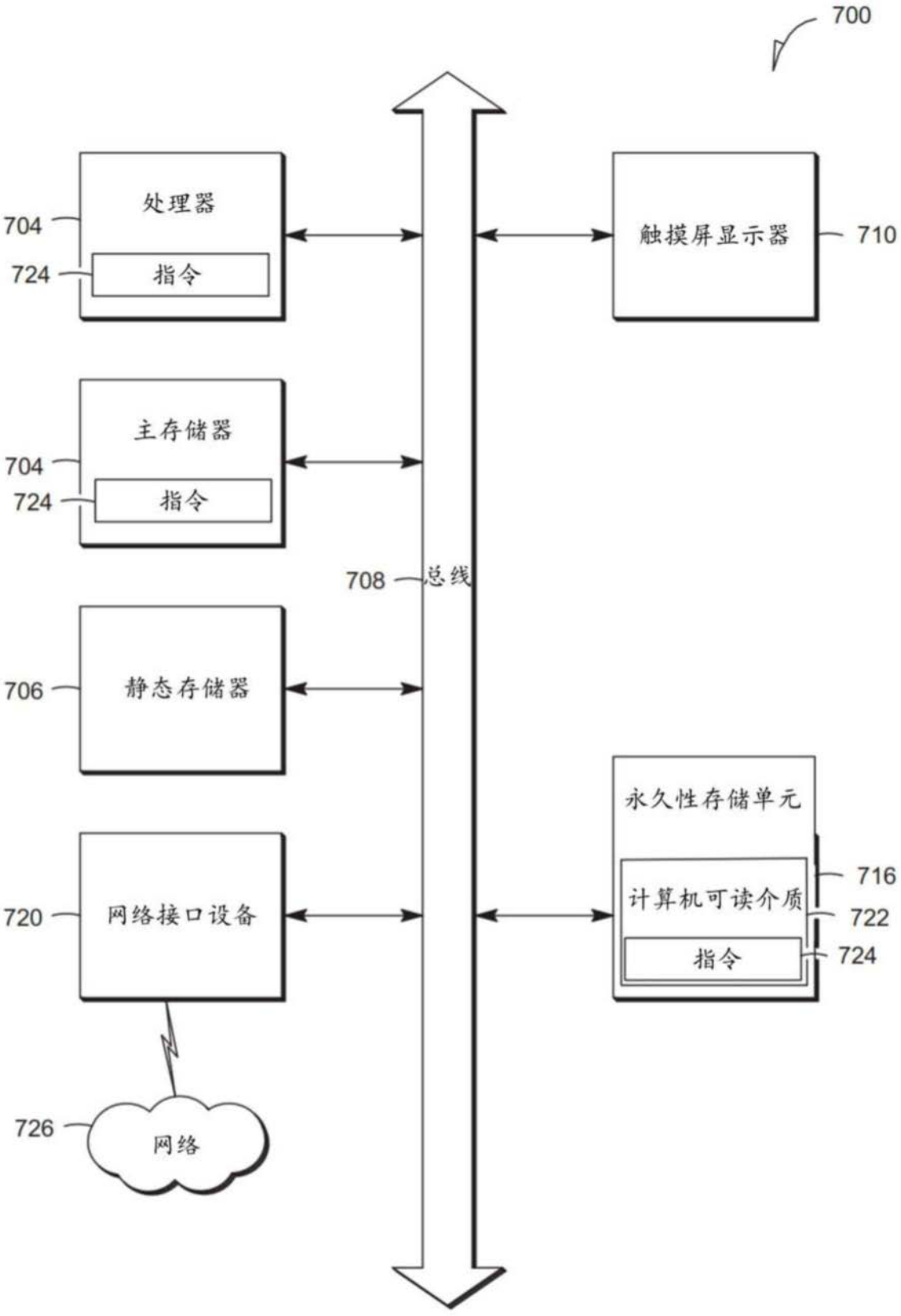

700
处理器
704
724
指令
触摸屏显示器
710
主存储器
704
724
指令
708
总线
静态存储器
706
永久性存储单元
716
计算机可读介质
722
指令
724
网络接口设备
720
726
网络

图7