

零信任安全容器的数据管控

技术领域

[01]本公开主要涉及小程序安全，尤其涉及小程序容器安全。

背景技术

[02]当前网络用户已大量地使用小程序。但随着小程序数量的爆发式增长，其安全风险也逐步显现。小程序 API 使用不规范、与第三方服务器业务逻辑交互有漏洞、服务器数据校验不严谨等情况都可能导致数据泄露。

[03]安全性成为了用户使用容器技术和业务上云面临的最大挑战，其中数据安全问题最为突出，近年来数据泄露事件发生的数量和泄露的数据量快速增长。

[04]因此，本领域需要能够一种对小程序容器的数据进行高效的严格管控的方案。

发明内容

[05]为解决上述技术问题，本公开提供了一种零信任安全容器的数据管控方案，该方案能够构建零信任安全容器的数据池，不仅基于零信任安全策略对进出数据池的数据进行验证，而且依据零信任安全策略对数据池内的数据进行隔离，从而实现对小程序容器的数据的严格管控。

[06]在本公开一实施例中，提供了一种零信任安全容器的数据管控方法，包括：依据场景生成零信任安全策略；构建零信任安全容器的数据池；针对要出入该数据池的数据采取零信任安全策略，其中要出入该数据池的数据在根据零信任安全策略验证后才被准入准出；以及基于零信任安全策略分层级隔离该数据池内的数据，以使得该数据池内的数据能够被定向读取。

[07]在本公开另一实施例中，零信任安全策略包括不信任任何数据读写方。

[08]在本公开另一实施例中，零信任安全策略包括：针对要写入该数据池的数据，验证来源；以及准入来源得到验证的数据，其中准入数据贴有来源标签。

[09]在本公开又一实施例中，零信任安全策略包括：针对要写入该数据池的数据，验证来源和数据写入方的角色；以及准入来源和数据写入方的角色得到验证的数据，其中准入数据贴有来源标签和数据写入方的角色标签。

[10]在本公开另一实施例中，零信任安全策略包括：针对要从该数据池读取的数据，验证数据读取方提供的来源、数据写入方的角色和数据读取方的角色；以及准出来源、数据写入方的角色和数据读取方的角色得到验证的数据。

[11]在本公开又一实施例中，依据零信任安全策略分层级隔离该数据池内的数据包括将该数据池隔离成多个数据域。

[12]在本公开另一实施例中，将该数据池隔离成多个数据域包括按照敏感数据和运行时数据将数据隔离到多个数据域中。

[13]在本公开另一实施例中，将该数据池隔离成多个数据域包括按照来源和角色将数据隔离到多个数据域中。

[14]在本公开又一实施例中，该多个数据域被设置相应的保护等级。

[15]在本公开一实施例中，提供了一种零信任安全容器的数据管控系统，包括：零信任策略模块，依据场景生成零信任安全策略；数据池构建模块，构建零信任安全容器的数据池；零信任验证模块，针对要出入该数据池的数据采取零信任安全策略，其中要出入该数据池的数据在根据零信任安全策略验证后才被准入准出；以及数据隔离模块，基于零信任安全策略分层级隔离该数据池内的数据，以使得该数据池内的数据能够被定向读取。

[16]在本公开另一实施例中，零信任安全策略包括不信任任何数据读写方。

[17]在本公开另一实施例中，零信任安全策略包括：针对要写入该数据池的数据，验证来源；以及准入来源得到验证的数据，其中准入数据贴有来源标签。

[18]在本公开又一实施例中，零信任安全策略包括：针对要写入该数据池的数据，验证来源和数据写入方的角色；以及准入来源和数据写入方的角色得到验证的数据，其中准入数据贴有来源标签和数据写入方的角色标签。

[19]在本公开另一实施例中，零信任安全策略包括：针对要从该数据池读取的数据，验证数据读取方提供的来源、数据写入方的角色和数据读取方的角色；以及准出来源、数据写入方的角色和数据读取方的角色得到验证的数据。

[20]在本公开又一实施例中，数据隔离模块基于零信任安全策略分层级隔离该数据池内的数据包括数据隔离模块将该数据池隔离成多个数据域。

[21]在本公开另一实施例中，数据隔离模块将该数据池隔离成多个数据域包括数据隔离模块按照敏感数据和运行时数据将数据隔离到多个数据域中。

[22]在本公开另一实施例中，数据隔离模块将该数据池隔离成多个数据域包括数据隔离模块按照来源和角色将数据隔离到多个数据域中。

[23]在本公开又一实施例中，多个数据域被设置相应的保护等级。

[24]在本公开一实施例中，提供了一种存储有指令的计算机可读存储介质，当这些指令被执行时使得机器执行如前所述的方法。

[25]提供本概述以便以简化的形式介绍以下在详细描述中进一步描述的一些概念。本概述并不旨在标识所要求保护主题的关键特征或必要特征，也不旨在用于限制所要求保护主题的范围。

附图说明

[26]本公开的以上发明内容以及下面的具体实施方式在结合附图阅读时会得到更好的理解。需要说明的是，附图仅作为所请求保护的发明的示例。在附图中，相同的附图标记代表相同或类似的元素。

[27]图 1 是示出根据本公开一实施例的零信任安全容器的数据管控方法的流程图；

[28]图 2 是示出根据本公开一实施例的对零信任安全容器进行数据管控的过程的示意图；

[29]图 3 是示出根据本公开一实施例的对零信任安全容器的数据管控能够达成的风险隔离的示意图；

[30]图 4 是示出根据本公开一实施例的零信任安全容器的数据池架构的示意图；

[31]图 5 是示出根据本公开一实施例的零信任安全容器的数据管控系统的框图。

具体实施方式

[32]使得本公开的上述目的、特征和优点能更加明显易懂，以下结合附图对本公开的具体实施方式作详细说明。

[33]在下面的描述中阐述了很多具体细节以便于充分理解本公开，但是本公开还可以采用其它不同于在此描述的其它方式来实施，因此本公开不受下文公开的具体实施例的限制。

[34]数据在整个生命周期有三种状态：静态（At-Rest）、传输中（In-Transit）和使用中（In-Use）。在静态状态下，一般会把数据存放在硬盘、闪存或其他存储设备中。传输中状态是指通过公网或私网将数据从一个地方传输到其他地方，用户可以在传输之前对文件加密或者采用安全的传输协议保证数据在传输中的安全，比如 HTTPS, SSL, TLS, FTPS 等。而使用中状态的数据在现有技术中的良好保护相对偏少。

[35]容器技术目前的形式主要体现在应用程序容器化（例如 Docker）和系统容器化（例如 LXC）中。这两种形式的容器均能让 IT 人员从底层架构中抽象出程序代码，从而实现跨各种部署环境的可移植性。容器的安全性通过恶意应用的隔离，来防止其对其他应用的破坏。主要的应用场景有：不可信负载隔离、多租户应用隔离、以及性能和故障隔离等。

[36]实际上，容器是一个特殊的进程，其通过名称空间（Namespace）、控制组（Control

groups)、切根(chroot)技术将资源、文件、设备、状态和配置划分到独立的空间。就应用程序容器化(即小程序容器)而言,虽然在框架上将小程序与平台方应用进行了充分的隔离,但是在数据的输入输出上几乎没有做任何管控,因此导致了大量越权、信息泄漏、隐私、数据安全、生态问题等等。

[37]容器当前对数据的设计模式是“宽进宽出”,即所有来源输入放入一个共享输入池,读写均没有身份验证等安全操作,根据key可以随意读写。返回数据也无校验,任意调用方都可以收到相同的返回。输入不做验证会导致攻击者可以利用输入容器的参数作为攻击载荷,干扰容器机制的正常运行。而输出不做验证会导致各种隐私泄漏风险。

[38]因此,要确保容器的安全性,还需要对数据(尤其是使用中数据)进行有效管控。本公开纳入零信任安全容器来实现对安全容器的严格数据管控。

[39]本公开将主要以Android系统的小程序容器为例进行安全容器的数据管控方案的具体描述,但本领域技术人员可以理解,本公开的技术方案同样适用于iOS系统的小程序容器,还适用于系统容器化的数据管控,在下文中将不再赘述。

[40]图1是示出根据本公开一实施例的零信任安全容器的数据管控方法100的流程图。

[41]在102,依据场景生成零信任安全策略。

[42]零信任安全是指在默认情况下不应该信任网络内部和外部的任何人、设备或系统,需要基于认证和授权重构访问控制的信任基础。诸如IP地址、主机、地理位置、所处网络等均不能作为可信的凭证。零信任本质上是以身份为中心进行访问控制,从而引导安全体系架构从“网络中心化”走向“身份中心化”。

[43]本公开的零信任安全容器对于所有的输入数据都不信任,不管其来自应用平台、小程序还是其他bundle。在Android系统中,bundle主要用于传递数据,它保存的数据是以key-value(键值对)的形式存在的。数据需要经过多重验证后才被准入。

[44]同样,本公开的零信任安全容器对于所有索要数据输出的读取方都不信任,不论该读取方是应用平台、小程序还是其他bundle。数据需要经过多重验证后才被准出。

[45]数据被准入准出之前的多重验证是基于零信任安全策略进行的。零信任安全策略的断言包括:应该始终假设充满威胁;容器外部、小程序、甚至容器内部每时每刻都充斥着威胁;不能仅仅凭借标签或参数来建立信任关系。零信任安全策略实际上可作为访问控制策略的底层设计,在不同的应用场景下访问控制策略可在零信任安全策略的断言的基础上因变于尽量多的数据源进行动态的评估和判定。

[46]在不同场景中,零信任安全策略可依据以上的原则进行调整。由此,可依据场景生成相应的零信任安全策略。

[47]在 104，构建零信任安全容器的数据池。

[48]要针对零信任安全容器进行数据管控，首先构建该零信任安全容器的数据池。该数据池的输入端、输出端和内部都将基于零信任安全进行数据管控。所有容器需要的敏感数据和运行时数据都被纳入数据池，并且在输入、输出和内部运作时基于零信任安全策略进行验证。

[49]该零信任安全容器的数据池的架构将参照图 4 在下文中具体描述。

[50]在 106，针对要出入数据池的数据采取零信任安全策略，其中该数据在根据零信任安全策略验证后才被准入准出。

[51]要对出入数据池的数据进行管控，需要采取零信任安全策略。所采取的零信任安全策略的断言包括：应该始终假设充满威胁；容器外部、小程序、甚至容器内部每时每刻都充斥着威胁；不能仅仅凭借标签或参数来建立信任关系。

[52]上述的零信任安全策略实质上包括不信任任何数据读写方。基于以上断言，对零信任安全容器所要使用的任何数据进行根据零信任安全策略的验证。针对零信任安全容器，数据的验证涉及验证发起方和验证操作方。验证发起方包括其他 bundle/SDK、容器内部、小程序、网络流量等；而验证操作方包括平台触发、外部网路触发、小程序自主触发、用户自主触发等。

[53]上述的零信任安全策略针对要使用的数据，将具体化为访问控制策略。针对零信任安全容器，在使用数据时，访问控制策略采用基于角色的访问控制（Role-based Access Control, RBAC），即角色决定权限，基于角色进行权限分层并采取最小权限原则。RBAC 单纯地将使用者和权限进行解耦，将使用者与角色、角色与权限关联。角色是为许多拥有相似权限的使用者进行的分类管理，角色具有上下级关系，可形成树状结构，父级角色的权限是自身以及子角色权限的总和。

[54]在本公开一实施例中，针对零信任安全容器的访问控制策略进一步包括针对要写入数据池的数据，验证数据的来源；以及准入来源得到验证的数据，其中准入数据贴有来源标签。在本公开一实施例中，数据的来源可包括网络、bundle、容器自身、jsapi 等。本领域技术人员可以理解，数据的来源可包括非穷尽的其他来源，在此不做赘述。

[55]在本公开另一实施例中，针对零信任安全容器的访问控制策略进一步包括针对要写入数据池的数据，验证数据的来源和数据写入方的角色；以及准入来源和数据写入方的角色得到验证的数据，其中准入数据贴有来源标签和数据写入方的角色标签。

[56]在本公开又一实施例中，针对零信任安全容器的访问控制策略还包括针对要从数据池读取的数据，验证数据读取方提供的来源、数据写入方的角色和数据读取方的角色；

以及准出来源、数据写入方的角色和数据读取方的角色得到验证的数据。

[57]该针对零信任安全容器的访问控制策略是自适应的，即利用机器学习来设置上下文相关访问策略，自动调整并适应该策略。

[58]在数据使用时，数据使用的方向性是明确的，即数据池内的数据能够被定向读取。在验证发起方和验证操作方被明确确定时，指定数据能够被访问。

[59]本领域技术人员可以理解，上述的零信任安全策略还可具体化为其他访问控制策略，例如自主访问控制（Discretionary Access Control, DAC）、强制访问控制（Mandatory Access Control, MAC）、Bell-Lapadula 安全模型、以及 Biba 安全模型等等。进一步地，这些访问控制策略可依据不同的应用场景针对不同层级的数据进行组合配置。在此对访问控制策略的设置不作赘述。

[60]在 108，基于零信任安全策略分层级隔离数据池内的数据，以使得该数据池内的数据能够被定向读取。

[61]Linux 容器采用 Namespace 来实现“隔离”。Namespace 修改了应用进程看待整个计算机的“视图”，即应用进程只能“看到”某些指定的内容。而对于宿主机来说这些被“隔离”了的进程跟其他进程并没有区别。

[62]由于使用 Namespace 作为隔离手段的容器并不需要单独的 Guest OS，这使得容器额外的资源占用几乎可以忽略不计；然而，这也造成了隔离并不彻底的问题，因为多个容器之间使用的还是同一宿主机的操作系统内核。进一步地，Linux 内核中的很多资源和对象是不能被 Namespace 化的，例如时间。

[63]因此采用 Linux Cgroups 来限制一个进程组能够使用的资源上限，包括 CPU、内存、磁盘、网络带宽等等。另外，Cgroups 还能够对进程进行优先级设置、审计，以及将进程挂起和恢复等操作。

[64]容器的以上隔离仅仅是在框架上将小程序与平台方应用隔离开，对于数据池内的数据并未进行隔离。

[65]针对数据池内的数据，可进行不同层级的管控以将数据安全隔离。从不同维度可对数据进行粗粒度和细粒度的管控。在安全容器的数据管控中，仅对敏感数据进行管控为粗粒度的管控。而不仅管控敏感数据、而且管控运行时数据，则为细粒度的管控。

[66]如图 2 所示，将在其他维度分层级对数据池内的数据进行隔离。即，在粗粒度层面上，对服务器之间的数据流做了访问控制，例如将不同来源的数据作隔离。而在细粒度层面上，基于角色（或身份，在本公开中概称为“角色”）和来源等来限制数据访问。对数据池内的每一个数据加上角色和来源标签，具有不同角色标签和来源标签的数据都

相互隔离。

[67]为了可以进一步提升数据访问的确定性，需要对角色在访问生命周期进行持续的动态验证，尤其是当上下文有变化的情况下，例如环境上下文、时空上下文、操作上下文、路径依赖等等。在零信任安全容器中，不信任任何数据读写方的零信任安全策略促使对使用数据的角色进行持续的动态验证，由此提高数据的安全性。

[68]在本公开一实施例中，只有角色和来源标签明确的数据才能够被写入数据池；而数据读取方需要携带自身角色以及目标角色才能准确读取数据。由此，该数据池内的数据能够被定向读取。

[69]因此，本公开所揭示的零信任安全容器的数据管控方法对进出数据池的数据进行验证，并对数据池内的数据进行隔离，由此确保零信任安全容器的数据安全。亦即，通过对容器所要使用的所有数据进行更细粒度的管控来实现对安全容器的严格数据管控。

[70]图 2 是示出根据本公开一实施例的对零信任安全容器进行数据管控的过程的示意图。本公开的零信任安全容器的数据池对于所有数据，都进行准入准出验证、隔离存储以及定向读取和使用。

[71]如图 2 所示，在本公开一实施例中，数据在进入数据池之前，将进行来源和数据写入方角色的验证。来源和数据写入方角色得到验证的数据被准入；相反来源或数据写入方角色不明确或得不到验证的数据不被准入。被准入的数据都被加上来源标签和数据写入方角色标签。

[72]而在本公开另一实施例中，数据在进入数据池之前，将进行来源的验证。来源得到验证的数据被准入；相反来源不明确或得不到验证的数据不被准入。被准入的数据都被加上来源标签。

[73]在本公开又一实施例中，数据在进入数据池之前，将进行数据写入方角色的验证。数据写入方角色得到验证的数据被准入；相反数据写入方角色不明确或得不到验证的数据不被准入。被准入的数据都被加上数据写入方角色标签。

[74]针对数据池内的数据，可进行不同层级的管控以将数据安全隔离。从不同维度可对数据进行粗粒度和细粒度的管控。在安全容器的数据管控中，仅对敏感数据进行管控为粗粒度的管控。而不仅管控敏感数据、而且管控运行时数据，则为细粒度的管控。

[75]如图 2 所示，则从数据的来源、数据写入方角色、数据读取方角色的维度来进行管控。在本公开一实施例中，数据可进行粗粒度的管控。例如，根据数据来源对数据进行分类或分级，形成不同数据域。数据将加上来源标签。又例如，根据数据读写方角色对数据进行分类或分级，形成不同数据域。数据将加上角色标签。在根据数据来源形成不

同数据域的情形中，可按来自不同的 web 服务器、应用服务器、数据库等来隔离数据，以使得源自不同服务器的数据能够分区域存储。并且，进一步地，web 服务器只能访问对应的应用服务器的数据，而应用服务器只能访问对应的数据库的数据。

[76]在本公开另一实施例中，数据可进行细粒度的管控。举例而言，根据数据来源和数据读写方角色来对数据进行分类分级，形成不同的数据域。具体地，对数据池内的每一个数据都加上角色和来源标签，具有不同角色标签和来源标签的数据都相互隔离，从而实现按角色和来源标签的数据隔离。

[77]在数据池内的数据分不同数据域隔离后，各个数据域可设置相应的保护等级这意味着数据在生成之后，在其存储、使用和传输的整个生命周期都根据对应的安全策略提供不同强度的安全防护。

[78]要获取数据池内的数据，在本公开一实施例中，同样读取方在指明双标签（即角色标签和来源标签）的情况下才可读取。读取方需要携带读取方角色（自身身份）以及数据写入方角色（目标身份）进行读取。举例而言，功能开发者需要知道自己使用的参数的上游标签才能准确获取参数。对于数据而言，需要经过角色和来源的验证才可被准出。由此，数据可以实现定向读取，即能够指明双标签的读取方可以读取数据。

[79]而在本公开另一实施例中，读取方在指明来源标签的情况下才可读取。对于数据而言，需要经过来源的验证才可被准出。由此，数据可以实现定向读取，即能够指明来源标签的读取方可以读取数据。

[80]在本公开又一实施例中，读取方在指明数据写入方角色情况下才可读取。对于数据而言，需要经过数据写入方角色的验证的数据才可被准出。由此，数据可以实现定向读取，即能够指明数据写入方角色标签的读取方可以读取数据。

[81]图 3 是示出根据本公开一实施例的对零信任安全容器的数据管控能够达成的风险隔离的示意图。

[82]根据本公开一实施例，由于如上所述的对零信任安全容器的数据管控，数据准入时的验证将使得该数据无法被利用作为攻击载荷，数据准出时的验证将阻止内部数据被外部窃取、并防止各种隐私泄露风险以保证数据安全，而数据的定向读取则可防止外部数据对内部机制运行的干扰，从而防止安全漏洞。

[83]在本公开中，由于采用细粒度的数据管控，将敏感数据和运行时数据都纳入了管控，因此才使得数据与风险相隔离、安全得到保障。

[84]图 4 是示出根据本公开一实施例的零信任安全容器的数据池架构的示意图。

[85]如图 4 所示，根据本公开一实施例，在 Android 系统的容器核心组件中，采用

Namespace 实现“进程隔离”，而采用 Cgroups 实现“权限隔离”。容器核心组件的运行参数和启动参数作为运行时数据进入容器数据中心。

[86]容器数据中心的数据池为零信任安全容器共享。任意容器中需要使用的外部输入参数，都只能从该数据池中获取，比如白名单、启动参数、jsapi 参数、开关配置等。

[87]容器数据中心的数据池配置有入口海关和出口海关，以对进出数据池的数据分别进行验证。数据池的数据来自容器核心组件、调用来源、容器依赖组件等等。容器核心组件的运行参数和启动参数、来自调用来源的数据、以及容器依赖组件的文件数据和用户隐私数据等等通过入口海关进入数据池。

[88]数据准入是写入数据前验证数据的过程，只有经过身份校验的数据才允许被写入池中。在本公开一实施例中，如图 4 所示，数据的身份校验可以是来源和角色双校验。进入数据池中的每一个数据都带有两个标签，即角色标签和来源标签。在本公开另一实施例中，数据的身份校验可以是来源或角色之一的校验。

[89]数据准出是输出数据前验证数据读取方的过程。在本公开一实施例中，读取方需要携带自身身份以及目标身份才能准确读取数据，且读取行为会被记录和管控。

[90]在本公开一实施例中，如图 4 所示，在容器数据中心的数据池中，不同角色标签和来源标签的参数相互隔离，并存储在不同的数据域中，例如第一数据域、第二数据域、第三数据域、第四数据域等。在这些数据域中，文件数据、内存数据、网络数据等等可分开存储。

[91]在该实施例中，这些数据域中的数据只有指明双标签的情况下才可读写，亦即功能开发者需要知道自己使用的参数的上游标签才能准确获取参数。该两类标签的生成由容器自身代码产生，不由读写的发起方生成。容器根据数据的来源（即，网络/ 其他 bundle/ 容器自身/jsapi）自主判断准入数据的来源标签和角色来源。

[92]对生成的角色标签和来源标签需要建立对应的资产库以及基于角色标签和来源标签的管控能力，在此不做赘述。

[93]本领域技术人员可以理解，如图 4 所示的零信任安全容器的数据池架构仅作为示例而非限制，数据的隔离可按图 2 所示依据不同维度根据场景来不同层级地进行。数据的标签也可根据场景的不同来不同地施加，并进行相应不同的准入、准出和隔离。

[94]图 5 是示出根据本公开一实施例的零信任安全容器的容器数据管控系统 500 的框图。

[95]容器数据管控系统 500 包括零信任策略模块 502、数据池构建模块 504、零信任验证模块 506 以及数据隔离模块 508。

[96]零信任策略模块 502 依据场景生成零信任安全策略。

[97]本公开的零信任安全容器对于所有的输入数据都不信任，不管其来自应用平台、小程序还是其他 bundle。在 Android 系统中，bundle 主要用于传递数据，它保存的数据是以 key-value（键值对）的形式存在的。数据需要经过多重验证后才被准入。同样，本公开的零信任安全容器对于所有索要数据输出的读取方都不信任，不论该读取方是应用平台、小程序还是其他 bundle。数据需要经过多重验证后才被准出。

[98]数据被准入准出之前的多重验证是基于零信任安全策略进行的。零信任安全策略的断言包括：应该始终假设充满威胁；容器外部、小程序、甚至容器内部每时每刻都充斥着威胁；不能仅仅凭借标签或参数来建立信任关系。零信任安全策略实际上可作为访问控制策略的底层设计。

[99]在不同场景中，零信任安全策略可依据以上的原则进行调整。由此，零信任策略模块 502 可依据场景生成相应的零信任安全策略。零信任策略模块 502 依据场景所生成的零信任安全策略被传递给零信任验证模块 506 和数据隔离模块 508。

[100]数据池构建模块 504 构建零信任安全容器的数据池。

[101]基于图 4 所示的架构，要针对零信任安全容器进行数据管控，数据池构建模块 504 首先构建该零信任安全容器的数据池。该数据池的输入端、输出端和内部都将基于零信任安全进行数据管控。所有容器需要的敏感数据和运行时数据都被纳入数据池，并且在输入、输出和内部运作时基于零信任安全策略进行验证。

[102]零信任验证模块 506 针对要出入数据池的数据采取零信任安全策略，其中要出入数据池的数据在根据零信任安全策略验证后才被准入准出。

[103]敏感数据和运行时数据将进入数据池构建模块 504 所构建的容器的共享数据池。要对出入数据池的数据进行管控，零信任验证模块 506 需要采取零信任安全策略。所采取的零信任安全策略的断言包括：应该始终假设充满威胁；容器外部、小程序、甚至容器内部每时每刻都充斥着威胁；不能仅仅凭借标签或参数来建立信任关系。上述的零信任安全策略实质上包括不信任任何数据读写方。

[104]上述的零信任安全策略针对要使用的数据，将具体化为访问控制策略。在本公开一实施例中，零信任验证模块 506 针对零信任安全容器的访问控制策略进一步包括针对要写入数据池的数据，验证数据的来源；以及准入来源得到验证的数据，其中准入数据贴有来源标签。

[105]在本公开另一实施例中，零信任验证模块 506 针对零信任安全容器的访问控制策略进一步包括针对要写入数据池的数据，验证数据的来源和数据写入方的角色；以及准入来源和数据写入方的角色得到验证的数据，其中准入数据贴有来源标签和数据写入方的

角色标签。

[106]在本公开又一实施例中，零信任验证模块 506 针对零信任安全容器的访问控制策略还包括针对要从数据池读取的数据，验证数据读取方提供的来源、数据写入方的角色和数据读取方的角色；以及准出来源、数据写入方的角色和数据读取方的角色得到验证的数据。

[107]数据隔离模块 508 基于零信任安全策略分层级隔离数据池内的数据，以使得数据池内的数据能够被定向读取。

[108]针对数据池内的数据，数据隔离模块 508 可进行不同层级的管控以将数据安全隔离。从不同维度可对数据进行粗粒度和细粒度的管控。在安全容器的数据管控中，仅对敏感数据进行管控为粗粒度的管控。而不仅管控敏感数据、而且管控运行时数据，则为细粒度的管控。

[109]如图 2 所示，数据隔离模块 508 将在其他维度分层级对数据池内的数据进行隔离。即，在粗粒度层面上，对服务器之间的数据流做了访问控制，例如将不同来源的数据作隔离。而在细粒度层面上，基于角色（或身份，在本公开中概称为“角色”）和来源等来限制数据访问。对数据池内的每一个数据加上角色和来源标签，具有不同角色标签和来源标签的数据都相互隔离。

[110]由此，数据隔离模块 508 基于零信任安全策略分层级隔离数据池内的数据之后，将使得数据池内的数据能够被定向调用。即，在数据使用时，数据使用的方向性是明确的，即数据池内的数据能够被定向读取。

[111]因此，本公开所揭示的零信任安全容器的数据管控系统对进出数据池的数据进行验证，并对数据池内的数据进行隔离，由此确保零信任安全容器的数据安全。亦即，通过对容器所要使用的所有数据进行更细粒度的管控来实现对安全容器的严格数据管控。

[112]以上描述的零信任安全容器的数据管控方法和系统的各个步骤和模块可以用硬件、软件、或其组合来实现。如果在硬件中实现，结合本发明描述的各种说明性步骤、模块、以及电路可用通用处理器、数字信号处理器（DSP）、专用集成电路（ASIC）、现场可编程门阵列（FPGA）、或其他可编程逻辑组件、硬件组件、或其任何组合来实现或执行。通用处理器可以是处理器、微处理器、控制器、微控制器、或状态机等。如果在软件中实现，则结合本发明描述的各种说明性步骤、模块可以作为一条或多条指令或代码存储在计算机可读介质上或进行传送。实现本发明的各种操作的软件模块可驻留在存储介质中，如 RAM、闪存、ROM、EPROM、EEPROM、寄存器、硬盘、可移动盘、CD-ROM、云存储等。存储介质可耦合到处理器以使得该处理器能从/向该存储介质读写信息，并执

行相应的程序模块以实现本发明的各个步骤。而且，基于软件的实施例可以通过适当的通信手段被上载、下载或远程地访问。这种适当的通信手段包括例如互联网、万维网、内联网、软件应用、电缆（包括光纤电缆）、磁通信、电磁通信（包括 RF、微波和红外通信）、电子通信或者其他这样的通信手段。

[113]还应注意，这些实施例可能是作为被描绘为流程图、流图、结构图、或框图的过程来描述的。尽管流程图可能会把诸操作描述为顺序过程，但是这些操作中有许多操作能够并行或并发地执行。另外，这些操作的次序可被重新安排。

[114]所公开的方法、装置和系统不应以任何方式被限制。相反，本发明涵盖各种所公开的实施例（单独和彼此的各种组合和子组合）的所有新颖和非显而易见的特征和方面。所公开的方法、装置和系统不限于任何具体方面或特征或它们的组合，所公开的任何实施例也不要求存在任一个或多个具体优点或者解决特定或所有技术问题。

[115]上面结合附图对本发明的实施例进行了描述，但是本发明并不局限于上述的具体实施方式，上述的具体实施方式仅仅是示意性的，而不是限制性的，本领域的普通技术人员在本发明的启示下，在不脱离本发明宗旨和权利要求所保护的范围情况下，还可做出很多更改，这些均落在本发明的保护范围之内。

权利要求书

- 1、一种零信任安全容器的数据管控方法，包括：
依据场景生成零信任安全策略；
构建零信任安全容器的数据池；
针对要出入所述数据池的数据采取所述零信任安全策略，其中要出入所述数据池的数据在根据所述零信任安全策略验证后才被准入准出；以及
基于所述零信任安全策略分层级隔离所述数据池内的数据，以使得所述数据池内的数据能够被定向读取。
- 2、如权利要求 1 所述的方法，所述零信任安全策略包括不信任任何数据读写方。
- 3、如权利要求 2 所述的方法，所述零信任安全策略包括：
针对要写入所述数据池的数据，验证来源；以及
准入所述来源得到验证的数据，其中所述准入数据贴有来源标签。
- 4、如权利要求 2 所述的方法，所述零信任安全策略包括：
针对要写入所述数据池的数据，验证来源和数据写入方的角色；以及
准入所述来源和所述数据写入方的角色得到验证的数据，其中所述准入数据贴有来源标签和所述数据写入方的角色标签。
- 5、如权利要求 3 所述的方法，所述零信任安全策略包括：
针对要从所述数据池读取的数据，验证数据读取方提供的来源、所述数据写入方的角色和所述数据读取方的角色；以及
准出所述来源、所述数据写入方的角色和所述数据读取方的角色得到验证的数据。
- 6、如权利要求 1 所述的方法，所述依据所述零信任安全策略分层级隔离所述数据池内的数据包括将所述数据池隔离成多个数据域。
- 7、如权利要求 6 所述的方法，所述将所述数据池隔离成多个数据域包括按照敏感数据和运行时数据将所述数据隔离到所述多个数据域中。
- 8、如权利要求 6 所述的方法，所述将所述数据池隔离成多个数据域包括按照来源和/或角色将所述数据隔离到所述多个数据域中。
- 9、如权利要求 6 所述的方法，所述多个数据域被设置相应的保护等级。
- 10、一种零信任安全容器的数据管控系统，包括：
零信任策略模块，依据场景生成零信任安全策略；
数据池构建模块，构建零信任安全容器的数据池；
零信任验证模块，针对要出入所述数据池的数据采取所述零信任安全策略，其中要

出入所述数据池的数据在根据所述零信任安全策略验证后才被准入准出；以及

数据隔离模块，基于所述零信任安全策略分层级隔离所述数据池内的数据，以使得所述数据池内的数据能够被定向读取。

11、如权利要求 10 所述的系统，所述零信任安全策略包括不信任任何数据读写方。

12、如权利要求 11 所述的系统，所述零信任安全策略包括：

针对要写入所述数据池的数据，验证来源；以及

准入所述来源得到验证的数据，其中所述准入数据贴有来源标签。

13、如权利要求 11 所述的系统，所述零信任安全策略包括：

针对要写入所述数据池的数据，验证来源和数据写入方的角色；以及

准入所述来源和所述数据写入方的角色得到验证的数据，其中所述准入数据贴有来源标签和所述数据写入方的角色标签。

14、如权利要求 11 所述的系统，所述零信任安全策略包括：

针对要从所述数据池读取的数据，验证数据读取方提供的来源、所述数据写入方的角色和所述数据读取方的角色；以及

准出所述来源、所述数据写入方的角色和所述数据读取方的角色得到验证的数据。

15、如权利要求 10 所述的系统，所述数据隔离模块基于所述零信任安全策略分层级隔离所述数据池内的数据包括所述数据隔离模块将所述数据池隔离成多个数据域。

16、如权利要求 15 所述的系统，所述数据隔离模块将所述数据池隔离成多个数据域包括所述数据隔离模块按照敏感数据和运行时数据将所述数据隔离到所述多个数据域中。

17、如权利要求 15 所述的系统，所述数据隔离模块将所述数据池隔离成多个数据域包括所述数据隔离模块按照来源和角色将所述数据隔离到所述多个数据域中。

18、如权利要求 15 所述的系统，所述多个数据域被设置相应的保护等级。

19、一种存储有指令的计算机可读存储介质，当所述指令被执行时使得机器执行如权利要求 1-9 中任一项所述的方法。