

DATA CONTROL FOR ZERO-TRUST SECURITY CONTAINER

TECHNICAL FIELD

[0001] This specification mainly relates to applet security, and in particular, to
5 applet container security.

BACKGROUND

[0002] Currently, network users use an extensive quantity of applets. However, as
the quantity of applets increases explosively, security risks of the applets gradually
emerge. Data leakage may be caused by improper use of applet APIs, vulnerabilities in
10 service logic interaction with third-party servers, and less rigorous server data
verification.

[0003] Security has become the biggest challenge for users to use container
technologies and cloud services. Data security is the most prominent problem. In recent
years, the quantity of data leaks and the amount of leaked data have increased rapidly.

15 [0004] Therefore, solutions for strictly and efficiently controlling data of applet
containers are needed in the industry.

SUMMARY

[0005] To resolve the above-mentioned technical problems, this specification
provides data control solutions for a zero-trust security container. In the solutions, a
20 data pool for a zero-trust security container can be built. Not only data to be in or out
of the data pool can be verified based on a zero-trust security policy, but also data in
the data pool can be isolated based on the zero-trust security policy, thereby
implementing strict control on data of applet containers.

[0006] According to some embodiments of this specification, a data control method
25 for a zero-trust security container is provided, including: a zero-trust security policy is
generated based on a scenario; a data pool for a zero-trust security container is built;

the zero-trust security policy is adopted for data to be in or out of the data pool, where admission or egress is allowed for the data to be in or out of the data pool only after the data is verified based on the zero-trust security policy; and data in the data pool is hierarchically isolated based on the zero-trust security policy, so that the data in the data pool can be read directionally.

[0007] In some other embodiments of this specification, the zero-trust security policy includes trusting none of data reading/writing parties.

[0008] In some other embodiments of this specification, the zero-trust security policy includes the following: a source is verified for data to be written to the data pool; and data whose source is verified is admitted, where the admitted data is labeled with a source label.

[0009] In still some other embodiments of this specification, the zero-trust security policy includes the following: a source and a role of a data writing party are verified for data to be written to the data pool; and data whose source and role of a data writing party are verified is admitted, where the admitted data is labeled with a source label and a role label of the data writing party.

[0010] In some other embodiments of this specification, the zero-trust security policy includes the following: a source, a role of a data writing party, and a role of a data reading party that are provided by the data reading party are verified for data to be read from the data pool; and data whose source, role of a data writing party, and role of a data reading party are verified is egressed.

[0011] In still some other embodiments of this specification, that data in the data pool is hierarchically isolated based on the zero-trust security policy includes the following: the data pool is isolated into multiple data domains.

[0012] In some other embodiments of this specification, that the data pool is isolated into multiple data domains includes the following: the data is isolated into the multiple data domains by sensitive data and runtime data.

[0013] In some other embodiments of this specification, that the data pool is isolated into multiple data domains includes the following: the data is isolated into the multiple data domains by source and/or role.

[0014] In still some other embodiments of this specification, corresponding protection levels are set for the multiple data domains.

[0015] According to some embodiments of this specification, a data control system for a zero-trust security container is provided, including: a zero-trust policy module,

configured to generate a zero-trust security policy based on a scenario; a data pool building module, configured to build a data pool for a zero-trust security container; a zero-trust verification module, configured to adopt the zero-trust security policy for data to be in or out of the data pool, where admission or egress is allowed for the data to be in or out of the data pool only after the data is verified based on the zero-trust security policy; and a data isolation module, configured to hierarchically isolate data in the data pool based on the zero-trust security policy, so that the data in the data pool can be read directionally.

[0016] In some other embodiments of this specification, the zero-trust security policy includes trusting none of data reading/writing parties.

[0017] In some other embodiments of this specification, the zero-trust security policy includes the following: a source is verified for data to be written to the data pool; and data whose source is verified is admitted, where the admitted data is labeled with a source label.

[0018] In still some other embodiments of this specification, the zero-trust security policy includes the following: a source and a role of a data writing party are verified for data to be written to the data pool; and data whose source and role of a data writing party are verified is admitted, where the admitted data is labeled with a source label and a role label of the data writing party.

[0019] In some other embodiments of this specification, the zero-trust security policy includes the following: a source, a role of a data writing party, and a role of a data reading party that are provided by the data reading party are verified for data to be read from the data pool; and data whose source, role of a data writing party, and role of a data reading party are verified is egressed.

[0020] In still some other embodiments of this specification, that the data isolation module hierarchically isolates data in the data pool based on the zero-trust security policy includes the following: the data isolation module isolates the data pool into multiple data domains.

[0021] In some other embodiments of this specification, that the data isolation module isolates the data pool into multiple data domains includes the following: the data isolation module isolates the data into the multiple data domains by sensitive data and runtime data.

[0022] In some other embodiments of this specification, that the data isolation module isolates the data pool into multiple data domains includes the following: the

data isolation module isolates the data into the multiple data domains by source and/or role.

[0023] In still some other embodiments of this specification, corresponding protection levels are set for the multiple data domains.

5 [0024] According to some embodiments of this specification, a computer-readable storage medium that stores an instruction is provided, where when the instruction is executed, a machine is enabled to perform the above-mentioned methods.

[0025] The summary is provided to describe some concepts in a simplified form. These concepts are further described in detail below. The summary is neither intended
10 to identify key features or essential features of the claimed subject matter nor is intended to limit the scope of the claimed subject matter.

DESCRIPTION OF THE DRAWINGS

[0026] The above-mentioned summary of this specification and the following specific implementations are better understood when read with reference to the
15 accompanying drawings. It should be noted that the accompanying drawings are merely examples of the claimed invention. The same reference numerals in accompanying drawings can represent same or similar elements.

[0027] FIG. 1 is a flowchart illustrating a data control method for a zero-trust security container, according to some embodiments of this specification;

20 [0028] FIG. 2 is a schematic diagram illustrating a process of performing data control on a zero-trust security container, according to some embodiments of this specification;

[0029] FIG. 3 is a schematic diagram illustrating risk isolation that can be achieved for data control on a zero-trust security container, according to some embodiments of
25 this specification;

[0030] FIG. 4 is a schematic diagram illustrating a data pool architecture for a zero-trust security container, according to some embodiments of this specification; and

[0031] FIG. 5 is a block diagram illustrating a data control system for a zero-trust security container, according to some embodiments of this specification.

DESCRIPTION OF EMBODIMENTS

[0032] The specific implementations of this specification will be described in detail below with reference to the accompanying drawings, so that the above-mentioned objectives, features, and advantages of this specification can be more clearly understood.

5 [0033] Many specific details are illustrated in the following description to facilitate adequate understanding of this specification. However, this specification can be implemented in another approach different from the approaches described here. Therefore, this specification is not limited to the specific embodiments disclosed below.

[0034] Data experiences three states throughout its lifecycle: at-rest, in-transit, and
10 in-use. In the at-rest state, the data is usually stored in a hard disk, a flash memory, or another storage device. The in-transit state means that the data is being transmitted from one place to another through a public network or a private network. A user can encrypt a file before transmission or use a secure transmission protocol to ensure data security during transmission, such as HTTPS, SSL, TLS, or FTPS. However, in conventional
15 technologies, the data in the in-use state is less well-protected.

[0035] Current forms of container technologies are mainly embodied in application containerization (for example, Docker) and system containerization (for example, LXC). Containers in both forms allow IT staff to abstract program code from an underlying architecture, thereby implementing portability across various deployment
20 environments. Security of the containers is achieved by isolating malicious applications to prevent them from destroying other applications. Main application scenarios include untrusted-load isolation, multi-tenant application isolation, and performance and fault isolation, etc.

[0036] In fact, a container is a special process that divides resources, files, devices, states, and configurations into separate spaces by using namespaces, control groups, and chroot technologies. With respect to application containerization (that is, an applet container), although applets are sufficiently isolated from platform-side applications in a framework, little control is performed on data inputs/outputs. Consequently, a large quantity of issues are caused, such as unauthorized access, information leakage, privacy
25 issues, data security issues, and ecological issues.

[0037] Currently, a designed mode of a container for data is "loose admission and loose egress", meaning that all source inputs are placed into one shared input pool, and security operations such as identity verification are not provided for reading/writing,

and random reading/writing can be performed based on a key. Moreover, return data is not verified, either, and any invoking party can receive the same return. Lack of verification on inputs may cause an attacker to use parameters of an input container as attack load, interfering with normal operation of a container mechanism. Lack of
5 verification on outputs may lead to various privacy leakage risks.

[0038] Therefore, to ensure container security, it is also necessary to perform effective control on data (especially in-use data). This specification incorporates a zero-trust security container to implement strict data control on security containers.

[0039] In this specification, an applet container of an Android system is mainly
10 used as an example for specific description of data control solutions for a security container. However, a person skilled in the art can understand that the technical solutions of this specification are also applicable to an applet container of an IOS system, and are further applicable to data control for system containerization. Details are omitted for simplicity below.

[0040] FIG. 1 is a flowchart illustrating a data control method 100 for a zero-trust security container, according to some embodiments of this specification.

[0041] In 102, a zero-trust security policy is generated based on a scenario.

[0042] Zero-trust security means that by default, no individual, device, or system
inside or outside a network should be trusted, and a trust foundation for access control
20 needs to be reconstructed based on authentication and authorization. For example, IP addresses, hosts, geographical locations, residing networks, etc. cannot be used as credentials of trustworthiness. Zero-trust is essentially identity-centric access control, leading a security system architecture from "network centralization" to "identity centralization".

[0043] The zero-trust security container of this specification trusts none of input
25 data, regardless of whether the data comes from application platforms, applets, or other bundles. In Android systems, a bundle is mainly used to transfer data. Data stored in the bundle exists in a key-value form (a key value pair). The data can be admitted only after multiple rounds of verification.

[0044] Similarly, the zero-trust security container of this specification trusts none
30 of reading parties that request data outputs, regardless of whether the reading parties are application platforms, applets, or other bundles. The data can be egressed only after multiple rounds of verification.

[0045] The multiple rounds of verification performed before data is admitted or egressed are based on the zero-trust security policy. Assertions of the zero-trust security policy include the following: It should be always assumed that threats are present; threats are present outside containers, in applets, and even inside the containers at all times; and trust relationships cannot be established only by using labels or parameters. The zero-trust security policy can actually serve as an underlying design of an access control policy. In different application scenarios, the access control policy can be dynamically evaluated and determined based on the assertions of the zero-trust security policy due to changes of as many data sources as possible.

[0046] In different scenarios, the zero-trust security policy can be adjusted based on the above-mentioned principles. Therefore, a corresponding zero-trust security policy can be generated based on a scenario.

[0047] In 104, a data pool for a zero-trust security container is built.

[0048] To perform data control on a zero-trust security container, a data pool for the zero-trust security container is first built. Data control is performed on all of an input end, an output end, and the inside of the data pool based on zero-trust security. All sensitive data and runtime data needed by the container are included into the data pool and verified based on the zero-trust security policy during input, output, and internal operation.

[0049] An architecture of the data pool for the zero-trust security container is described in detail below with reference to FIG. 4.

[0050] In 106, the zero-trust security policy is adopted for data to be in or out of the data pool, where admission or egress is allowed for the data only after the data is verified based on the zero-trust security policy.

[0051] The zero-trust security policy needs to be adopted to perform control on the data to be in or out of the data pool. Assertions of the adopted zero-trust security policy include the following: It should be always assumed that threats are present; threats are present outside containers, in applets, and even inside the containers at all times; and trust relationships cannot be established only by using labels or parameters.

[0052] The zero-trust security policy substantially includes trusting none of data reading/writing parties. Based on the above-mentioned assertions, any data to be used by the zero-trust security container is verified based on the zero-trust security policy. For the zero-trust security container, data verification involves a verification initiator and a verification operator. The verification initiator includes another bundle/SDK, an

interior of a container, an applet, network traffic, etc. The verification operator includes platform triggering, external network triggering, applet autonomous triggering, user autonomous triggering, etc.

[0053] The above-mentioned zero-trust security policy is specifically an access control policy for data to be used. For the zero-trust security container, during data use, the access control policy uses role-based access control (RBAC), in other words, a role determines a permission, and permissions are hierarchized based on roles, and a principle of minimum permissions is followed. The RBAC simply decouples a user from a permission, but associates a user with a role and the role with a permission. Roles are classification management performed for many users with similar permissions. The roles are in upper-lower relationships and can form a tree structure. Permissions of a parent role are the sum of permissions of the parent role itself and permissions of child roles.

[0054] In some embodiments of this specification, the access control policy for the zero-trust security container further includes the following: a data source is verified for data to be written to the data pool; and data whose source is verified is admitted, where the admitted data is labeled with a source label. In some embodiments of this specification, the data source can include a network, a bundle, a container itself, a JSAPI, etc. A person skilled in the art can understand that the data source can include other sources that are not exhausted, and details are omitted here for simplicity.

[0055] In some other embodiments of this specification, the access control policy for the zero-trust security container further includes the following: a source and a role of a data writing party are verified for data to be written to the data pool; and data whose source and role of a data writing party are verified is admitted, where the admitted data is labeled with a source label and a role label of the data writing party.

[0056] In still some other embodiments of this specification, the access control policy for the zero-trust security container further includes the following: a source, a role of a data writing party, and a role of a data reading party that are provided by the data reading party are verified for data to be read from the data pool; and data whose source, role of a data writing party, and role of a data reading party are verified is egressed.

[0057] The access control policy for the zero-trust security container is self-adaptive, that is, a context-related access policy is set by using machine learning, and the policy is automatically adjusted and adapted.

[0058] During data use, directivity of the data use is clear, that is, the data in the data pool can be read directionally. When the verification initiator and the verification operator are explicitly determined, specified data can be accessed.

[0059] A person skilled in the art can understand that, the above-mentioned zero-trust security policy can further be embodied as another access control policy, such as Discretionary Access Control (DAC), Mandatory Access Control (MAC), a Bell-Lapadula security model, or a Biba security model. Further, these access control policies can be combined and configured for different levels of data based on different application scenarios. Details about settings of the access control policy are omitted here for simplicity.

[0060] In 108, data in the data pool are hierarchically isolated based on the zero-trust security policy, so that the data in the data pool can be read directionally.

[0061] Linux containers use a Namespace to implement "isolation". The Namespace modifies a "view" of a whole computer perceived by an application process, that is, the application process can only "see" certain specified content. For a host, "isolated" processes are not different from other processes.

[0062] Because a container that uses a Namespace as an isolation means does not need a separate Guest OS, additional resource occupation by the container is almost negligible. However, such practice also poses a problem of incomplete isolation, as an operating system kernel of the same host is used by multiple containers. Further, many resources and objects in a Linux kernel cannot be Namespace-based, for example, time.

[0063] Therefore, Linux Cgroups is used to limit upper limits of resources that can be used by a process group, including a CPU, a memory, a disk, a network bandwidth, etc. In addition, Cgroups can further perform priority setting, auditing, and process suspension and restoration for processes.

[0064] The above-mentioned isolation of a container is merely to isolate applets from platform-side applications in the framework, and the data in the data pool is not isolated.

[0065] For the data in the data pool, different levels of control can be performed to securely isolate the data. Coarse-grained and fine-grained control can be performed on the data from different dimensions. In data control of a security container, only coarse-grained control is performed on sensitive data. In addition to control on sensitive data, fine-grained control is performed on runtime data.

[0066] As shown in FIG. 2, the data in the data pool is hierarchically isolated from other dimensions. To be specific, at a coarse-grained level, access control is performed on a data stream between servers, for example, data from different sources is isolated. At a fine-grained level, data access is restricted based on a role (or an identity, which is collectively referred to as a "role" here), a source, etc. Each piece of data in the data pool is added with a role label and a source label. Data with different role labels and source labels are mutually isolated.

[0067] To further improve certainty of data access, a role needs to be continuously and dynamically verified throughout an access lifecycle, especially in a case with changes of contexts, such as an environment context, a space-time context, an operation context, and path dependency. In the zero-trust security container, the zero-trust security policy that trusts none of data reading/writing parties promotes continuous dynamic verification on a role that is to use data, thereby improving data security.

[0068] In some embodiments of this specification, only data with an explicit role label and source label can be written to the data pool. A data reading party needs to carry its own role and a target role to accurately read data. Therefore, the data in the data pool can be read directionally.

[0069] Therefore, the data control method for a zero-trust security container disclosed in this specification verifies data to be in or out of the data pool, and isolates the data in the data pool, thereby ensuring data security of the zero-trust security container. In other words, strict data control on a security container is implemented by performing finer-grained control on all data to be used by the container.

[0070] FIG. 2 is a schematic diagram illustrating a process of performing data control on a zero-trust security container, according to some embodiments of this specification. The data pool for the zero-trust security container in this specification performs admission/egress verification, isolated storage, and directional reading and use for all data.

[0071] As shown in FIG. 2, in some embodiments of this specification, before data enters the data pool, a source and a role of a data writing party role are verified. Data whose source and role of a data writing party are verified is admitted. On the contrary, data whose source or role of a data writing party are unclear or cannot be verified is not admitted. All admitted data is added with a source label and a role label of a data writing party.

[0072] In some other embodiments of this specification, before data enters the data pool, a source is verified. Data whose source is verified is admitted. On the contrary, data whose source is unclear or cannot be verified is not admitted. All admitted data is added with a source label.

5 **[0073]** In still some other embodiments of this specification, before data enters the data pool, a role of a data writing party is verified. Data whose role of a data writing party is verified is admitted. On the contrary, data whose role of a data writing party is unclear or cannot be verified is not admitted. All admitted data is added with a role label of a data writing party.

10 **[0074]** For the data in the data pool, different levels of control can be performed to securely isolate the data. Coarse-grained and fine-grained control can be performed on the data from different dimensions. In data control of a security container, only coarse-grained control is performed on sensitive data. In addition to control on sensitive data, fine-grained control is performed on runtime data.

15 **[0075]** As shown in FIG. 2, control is performed from dimensions of a data source, a role of a data writing party, and a role of a data reading party. In some embodiments of this specification, coarse-grained control can be performed on data. For example, data is classified or hierarchized based on a data source to form different data domains. The data is added with a source label. For another example, data is classified or
20 hierarchized based on a role of a data reading/writing party to form different data domains. The data is added with a role label. When different data domains are formed based on data sources, data can be isolated based on different web servers, application servers, databases, etc., so that data from different servers can be stored in different domains. In addition, further, a web server can access only data of a corresponding
25 application server, and an application server can access only data of a corresponding database.

[0076] In some other embodiments of this specification, fine-grained control can be performed on data. For example, data is classified or hierarchized based on a data source and a role of a data reading/writing party to form different data domains.
30 Specifically, each piece of data in the data pool is added with a role label and a source label. Data with different role labels and source labels are mutually isolated, thereby implementing data isolation by role label and source label.

[0077] After the data in the data pool is isolated into different data domains, corresponding protection levels can be set for the data domains. Such practice means

that after data is generated, different levels of security protection are provided based on a corresponding security policy throughout a lifecycle of storage, use, and transmission of the data.

[0078] To obtain data in the data pool, in some embodiments of this specification, similarly, the data can be read by a reading party only when dual labels (that is, a role label and a source label) are specified. The reading party needs to carry a role of the reading party (its own identity) and a role of a data writing party (a target identity) to perform reading. For example, a function developer needs to know an upstream label of a parameter used by the function developer to accurately obtain the parameter. Data can be egressed only when a role and a source are verified. As such, data can be read directionally, that is, a reading party that can specify dual labels can read the data.

[0079] In some other embodiments of this specification, a reading party can read only when specifying a source label. Data can be egressed only when a source is verified. As such, data can be read directionally, that is, a reading party that can specify a source label can read the data.

[0080] In still some other embodiments of this specification, a reading party can read only when specifying a role of a data writing party. Only data whose role of a data writing party is verified can be egressed. As such, data can be read directionally, that is, a reading party that can specify a role label of the data writing party can read the data.

[0081] FIG. 3 is a schematic diagram illustrating risk isolation that can be achieved for data control on a zero-trust security container, according to some embodiments of this specification.

[0082] According to some embodiments of this specification, due to the above-mentioned data control on a zero-trust security container, verification upon data admission can prevent data from being exploited as attack load, verification upon data egress can prevent internal data from being stolen externally and prevent various privacy leakage risks to ensure data security, and directional reading of data can prevent external data from interfering with operation of an internal mechanism, thereby reducing security vulnerabilities.

[0083] In this specification, as fine-grained data control is performed on both sensitive data and runtime data, the data can be isolated from risks and security can be ensured.

[0084] FIG. 4 is a schematic diagram illustrating a data pool architecture for a zero-trust security container, according to some embodiments of this specification.

[0085] As shown in FIG. 4, according to some embodiments of this specification, in a container core component of an Android system, "process isolation" is implemented by using a Namespace, and "permission isolation" is implemented by using Cgroups. A running parameter and a startup parameter of the container core component are used as runtime data to enter a container data center.

[0086] A data pool of the container data center is shared by zero-trust security containers. An external input parameter that needs to be used in any container, such as a whitelist, a start parameter, a JSAPI parameter, or a switch configuration, can only be obtained from the data pool.

[0087] The data pool of the container data center is configured with an ingress gateway and an egress gateway to separately verify data to be in or out of the data pool. Data in the data pool comes from the container core component, an invoking source, a container dependency component, etc. The running parameter and the startup parameter of the container core component, data from the invoking source, file data and user privacy data of the container dependency component, etc. enter the data pool through the ingress gateway.

[0088] Data admission is a process of verifying data before the data is written. Only data whose identity is verified can be written to the pool. In some embodiments of this specification, as shown in FIG. 4, identity verification on data can be verification on both a source and a role. Each piece of data entering the data pool carries two labels, that is, a role label and a source label. In some other embodiments of this specification, identity verification on data can be verification on either a source or a role.

[0089] Data egress is a process of verifying a data reading party before data is output. In some embodiments of this specification, the reading party needs to carry its own identity and a target identity to accurately read data, and a reading behavior is recorded and controlled.

[0090] In some embodiments of this specification, as shown in FIG. 4, in the data pool of the container data center, parameters with different role labels and source labels are mutually isolated, and are stored in different data domains, for example, a first data domain, a second data domain, a third data domain, and a fourth data domain. In these data domains, file data, memory data, network data, etc. can be stored separately.

[0091] In the embodiments, data in these data domains can be read/written only when dual labels are specified, that is, a function developer needs to know an upstream label of a parameter used by the function developer to accurately obtain the parameter. The two types of labels are generated by a container's own code, and is not generated
5 by a read/write initiator. A container independently determines a source label and a role source of admitted data based on a data source (i.e., a network/another bundle/the container itself/a JSAPI).

[0092] For generated role labels and source labels, a corresponding asset library and a control capability based on the role labels and source labels need to be established,
10 details are omitted here for simplicity.

[0093] A person skilled in the art can understand that the data pool architecture for a zero-trust security container shown in FIG. 4 is merely an example rather than a limitation, and data isolation can be performed at different levels based on different dimensions and scenarios, as shown in FIG. 2. Data labels can also be applied
15 differentially based on different scenarios, and different admission, egress, and isolation can be performed accordingly.

[0094] FIG. 5 is a block diagram illustrating a container data control system 500 for a zero-trust security container, according to some embodiments of this specification.

[0095] The container data control system 500 includes a zero-trust policy module
20 502, a data pool building module 504, a zero-trust verification module 506, and a data isolation module 508.

[0096] The zero-trust policy module 502 generates a zero-trust security policy based on a scenario.

[0097] The zero-trust security container of this specification trusts none of input
25 data, regardless of whether the data comes from application platforms, applets, or other bundles. In Android systems, a bundle is mainly used to transfer data. Data stored in the bundle exists in a key-value form (a key value pair). The data can be admitted only after multiple rounds of verification. Similarly, the zero-trust security container of this specification trusts none of reading parties that request data outputs, regardless of
30 whether the reading parties are application platforms, applets, or other bundles. The data can be egressed only after multiple rounds of verification.

[0098] The multiple rounds of verification performed before data is admitted or egressed are based on the zero-trust security policy. Assertions of the zero-trust security policy include the following: It should be always assumed that threats are present;

threats are present outside containers, in applets, and even inside the containers at all times; and trust relationships cannot be established only by using labels or parameters. The zero-trust security policy can actually serve as an underlying design of an access control policy.

5 **[0099]** In different scenarios, the zero-trust security policy can be adjusted based on the above-mentioned principles. Therefore, the zero-trust policy module 502 can generate a corresponding zero-trust security policy based on a scenario. The zero-trust security policy generated by the zero-trust policy module 502 based on the scenario is transferred to the zero-trust verification module 506 and the data isolation module 508.

10 **[0100]** The data pool building module 504 builds a data pool for a zero-trust security container.

[0101] Based on the architecture shown in FIG. 4, to perform data control on the zero-trust security container, the data pool building module 504 first builds the data pool for the zero-trust security container. Data control is performed on all of an input
15 end, an output end, and the inside of the data pool based on zero-trust security. All sensitive data and runtime data needed by the container are included into the data pool and verified based on the zero-trust security policy during input, output, and internal operation.

[0102] The zero-trust verification module 506 adopts the zero-trust security policy
20 for data to be in or out of the data pool, where admission or egress is allowed for the data to be in or out of the data pool only after the data is verified based on the zero-trust security policy.

[0103] The sensitive data and runtime data are to enter the shared data pool for the container that is built by the data pool building module 504. The zero-trust verification
25 module 506 needs to adopt the zero-trust security policy to perform control on the data to be in or out of the data pool. Assertions of the adopted zero-trust security policy include the following: It should be always assumed that threats are present; threats are present outside containers, in applets, and even inside the containers at all times; and trust relationships cannot be established only by using labels or parameters. The zero-
30 trust security policy substantially includes trusting none of data reading/writing parties.

[0104] The above-mentioned zero-trust security policy is specifically an access control policy for data to be used. In some embodiments of this specification, the access control policy of the zero-trust verification module 506 for the zero-trust security container further includes the following: a data source is verified for data to be written

to the data pool; and data whose source is verified is admitted, where the admitted data is labeled with a source label.

5 **[0105]** In some other embodiments of this specification, the access control policy of the zero-trust verification module 506 for the zero-trust security container further includes the following: a source and a role of a data writing party are verified for data to be written to the data pool; and data whose source and role of a data writing party are verified is admitted, where the admitted data is labeled with a source label and a role label of the data writing party.

10 **[0106]** In still some other embodiments of this specification, the access control policy of the zero-trust verification module 506 for the zero-trust security container further includes the following: a source, a role of a data writing party, and a role of a data reading party that are provided by the data reading party are verified for data to be read from the data pool; and data whose source, role of a data writing party, and role of a data reading party are verified is egressed.

15 **[0107]** The data isolation module 508 hierarchically isolates data in the data pool based on the zero-trust security policy, so that the data in the data pool can be read directionally.

20 **[0108]** For the data in the data pool, the data isolation module 508 can perform different levels of control to securely isolate the data. Coarse-grained and fine-grained control can be performed on the data from different dimensions. In data control of a security container, only coarse-grained control is performed on sensitive data. In addition to control on sensitive data, fine-grained control is performed on runtime data.

25 **[0109]** As shown in FIG. 2, the data isolation module 508 hierarchically isolates the data in the data pool from other dimensions. To be specific, at a coarse-grained level, access control is performed on a data stream between servers, for example, data from different sources is isolated. At a fine-grained level, data access is restricted based on a role (or an identity, which is collectively referred to as a "role" here), a source, etc. Each piece of data in the data pool is added with a role label and a source label. Data with different role labels and source labels are mutually isolated.

30 **[0110]** Therefore, after the data isolation module 508 hierarchically isolates the data in the data pool based on the zero-trust security policy, the data in the data pool can be invoked directionally. In other words, during data use, directivity of the data use is clear, that is, the data in the data pool can be read directionally.

[0111] Therefore, the data control system for a zero-trust security container disclosed in this specification verifies data to be in or out of the data pool, and isolates the data in the data pool, thereby ensuring data security of the zero-trust security container. In other words, strict data control on a security container is implemented by performing finer-grained control on all data to be used by the container.

[0112] Steps and modules of the above-mentioned data control method and system for a zero-trust security container can be implemented by hardware, software, or a combination thereof. For implementation in hardware, various illustrative steps, modules, and circuits described with reference to this specification can be implemented or executed by a general-purpose processor, a digital signal processor (DSP), an application-specific integrated circuit (ASIC), a field programmable gate array (FPGA), or another programmable logic component, hardware component, or any combination thereof. The general-purpose processor can be a processor, a microprocessor, a controller, a microcontroller, or a state machine, etc. For implementation in software, various illustrative steps and modules described with reference to this specification can be stored on a computer-readable medium or transmitted as one or more instructions or code. A software module for implementing various operations of this specification can reside in a storage medium, such as a RAM, a flash memory, a ROM, an EPROM, an EEPROM, a register, a hard disk, a removable disk, a CD-ROM, or a cloud storage, etc. The storage medium can be coupled to a processor, so that the processor can read information from/write information to the storage medium, and execute a corresponding program module to implement the steps of this specification. Furthermore, software-based embodiments can be uploaded, downloaded, or remotely accessed by suitable communication means. Such suitable communication means include, for example, the Internet, the World Wide Web, intranets, software applications, cables (including optical fiber cables), magnetic communication, electromagnetic communication (including RF, microwaves, and infrared communication), electronic communication, or other such communication means.

[0113] It should further be noted that these embodiments may be described as processes depicted as flowcharts, flow diagrams, structural diagrams, or block diagrams. Although a flowchart may describe operations as sequential processes, many of these operations can be performed in parallel or concurrently. In addition, an order of these operations can be re-arranged.

[0114] The disclosed methods, apparatuses, and systems shall not be limited in any way. In contrast, this specification covers all novel and non-obvious features and aspects of various disclosed embodiments (individual embodiments and various combinations and sub-combinations of the embodiments). The disclosed methods, apparatuses, and systems are not limited to any particular aspect or feature or combination thereof, and any embodiment disclosed does not require presence of any particular advantage or advantages or solutions to specific or all technical problems.

[0115] The embodiments of this specification are described with reference to the accompanying drawings above. However, this specification is not limited to the above-mentioned specific implementations, which are merely examples but not limitations. A person of ordinary skill in the art under an enlightenment of this specification can make many modifications without departing from the objective of this specification and the protection scope of the claims. All of these modifications shall fall within the protection scope of this specification.

CLAIMS

What is claimed is:

1. A data control method for a zero-trust security container, comprising:
generating a zero-trust security policy based on a scenario;
5 building a data pool for a zero-trust security container;
adopting the zero-trust security policy for data to be in or out of the data pool,
wherein admission or egress is allowed for the data to be in or out of the data pool only
after the data is verified based on the zero-trust security policy; and
hierarchically isolating data in the data pool based on the zero-trust security policy,
10 so that the data in the data pool can be read directionally.
2. The method according to claim 1, wherein the zero-trust security policy
comprises trusting none of data reading/writing parties.
3. The method according to claim 2, wherein the zero-trust security policy
comprises the following:
15 verifying a source for data to be written to the data pool; and
admitting data whose source is verified, wherein the admitted data is labeled with
a source label.
4. The method according to claim 2, wherein the zero-trust security policy
comprises the following:
20 verifying a source and a role of a data writing party for data to be written to the
data pool; and
admitting data whose source and role of a data writing party are verified, wherein
the admitted data is labeled with a source label and a role label of the data writing party.
5. The method according to claim 3, wherein the zero-trust security policy
25 comprises the following:
verifying, for data to be read from the data pool, a source, a role of the data writing
party, and a role of a data reading party that are provided by the data reading party; and
egressing data whose source, role of a data writing party, and role of a data reading
party are verified.
- 30 6. The method according to claim 1, wherein the hierarchically isolating data in
the data pool based on the zero-trust security policy comprises isolating the data pool
into multiple data domains.

7. The method according to claim 6, wherein the isolating the data pool into multiple data domains comprises isolating the data into the multiple data domains by sensitive data and runtime data.

5 8. The method according to claim 6, wherein the isolating the data pool into multiple data domains comprises isolating the data into the multiple data domains by source and/or role.

9. The method according to claim 6, wherein corresponding protection levels are set for the multiple data domains.

10 10. A data control system for a zero-trust security container, comprising:
a zero-trust policy module, configured to generate a zero-trust security policy based on a scenario;

a data pool building module, configured to build a data pool for a zero-trust security container;

15 a zero-trust verification module, configured to adopt the zero-trust security policy for data to be in or out of the data pool, wherein admission or egress is allowed for the data to be in or out of the data pool only after the data is verified based on the zero-trust security policy; and

20 a data isolation module, configured to hierarchically isolate data in the data pool based on the zero-trust security policy, so that the data in the data pool can be read directionally.

11. The system according to claim 10, wherein the zero-trust security policy comprises trusting none of data reading/writing parties.

12. The system according to claim 11, wherein the zero-trust security policy comprises the following:

25 verifying a source for data to be written to the data pool; and

admitting data whose source is verified, wherein the admitted data is labeled with a source label.

13. The system according to claim 11, wherein the zero-trust security policy comprises the following:

30 verifying a source and a role of a data writing party for data to be written to the data pool; and

admitting data whose source and role of a data writing party are verified, wherein the admitted data is labeled with a source label and a role label of the data writing party.

14. The system according to claim 11, wherein the zero-trust security policy comprises the following:

verifying, for data to be read from the data pool, a source, a role of the data writing party, and a role of a data reading party that are provided by the data reading party; and

5 egressing data whose source, role of a data writing party, and role of a data reading party are verified.

15. The system according to claim 10, wherein that the data isolation module hierarchically isolates data in the data pool based on the zero-trust security policy comprises the following: the data isolation module isolates the data pool into multiple

10 data domains.

16. The system according to claim 15, wherein that the data isolation module isolates the data pool into multiple data domains comprises the following: the data isolation module isolates the data into the multiple data domains by sensitive data and runtime data.

15 17. The system according to claim 15, wherein that the data isolation module isolates the data pool into multiple data domains comprises the following: the data isolation module isolates the data into the multiple data domains by source and/or role.

18. The system according to claim 15, wherein corresponding protection levels are set for the multiple data domains.

20 19. A computer-readable storage medium that stores an instruction, wherein when the instruction is executed, a machine is enabled to perform the method according to any one of claims 1 to 9.