FILE PROTECTION METHOD, RULE CONFIGURATION METHOD, AND COMPUTING DEVICE

[0001] This application claims priority to Chinese Patent Application No.202311080613.6, filed with the China National Intellectual Property Administration on August 24, 2023, and entitled "FILE PROTECTION METHOD, RULE CONFIGURATION METHOD, AND COMPUTING DEVICE", which is incorporated herein by reference in its entirety.

TECHNICAL FIELD

[0002] The present application relates to the field of computer technologies, and in particular, to a file protection method, a rule configuration method, and a computing device.

BACKGROUND

[0003] With the arrival of a cloud native era, providing a safe and portable cloud service for a user by using a container technology becomes a development trend. How to protect a file in a container from being maliciously tampered also becomes a technical problem that needs to be resolved.

[0004] Although a current container management tool such as K8s (Kubernetes, an open-source container management platform) provides a resource isolation function between containers, so that an application program in one container cannot access a file in another container, only file protection between the containers can be implemented in this file protection manner, and a potential security risk still exists. Consequently, the container file cannot be effectively protected.

SUMMARY

[0005] Embodiments of the present application provide a file protection method, a rule configuration method, and a computing device, to resolve a technical problem in the prior art

that a container file has a potential security risk, and file protection cannot be effectively implemented.

[0006] According to a first aspect, an embodiment of the present application provides a file protection method, applied to a device driver in a kernel. The method includes:

detecting a file access request initiated by an application program;

determining a container identifier of a target container to which the application program belongs;

searching for a container label associated with the container identifier;

searching for at least one corresponding file defense rule according to the container label; and

determining, according to the at least one file defense rule, whether to perform a defense action on the file access request.

[0007] According to a second aspect, an embodiment of the present application provides a file protection method, applied to a device driver in a kernel. The method includes:

detecting a file access request initiated by an application program, and determining file information;

determining a container identifier of a target container corresponding to the application program;

sending the container identifier and the file information to a defense program, so that the defense program searches for a container label associated with the container identifier, searches for at least one corresponding file defense rule according to the container label, and generates indication information according to a matching result between the file information and the at least one file defense rule; and

determining, according to the indication information sent by the defense program, whether to perform a defense action on the file access request.

[0008] According to a third aspect, an embodiment of the present application provides a file protection method, applied to a defense program. The method includes:

obtaining a container identifier and file information that are sent by a device driver, where the file information is determined by the device driver by detecting a file access request initiated by an application program, and the container identifier is obtained through searching based on the application program;

searching for a container label associated with the container identifier;

searching for at least one corresponding file defense rule according to the container label;

generating indication information according to a matching result between the file information and the at least one file defense rule; and

sending the indication information to the device driver, where the indication information is used to indicate whether to perform a defense action on the file access request.

[0009] According to a fourth aspect, an embodiment of the present application provides a rule configuration method, including:

providing rule configuration prompt information on a rule configuration interface;

in response to a user configuration operation, determining a file defense rule configured by a user and a container label set for the file defense rule; and

sending the file defense rule and the container label to a defense program, so that the defense program determines an association relationship between the file defense rule and the container label;

where a device driver is configured to: detect a file access request initiated by an application program, and determine a container identifier of a target container corresponding to the application program, where the container identifier is used to determine an associated container label, the container label is used to determine at least one corresponding file defense rule by searching for the association relationship, and the at least one file defense rule is used to determine whether to perform a defense action on the file access request.

[0010] According to a fifth aspect, an embodiment of the present application provides a file protection method, including:

obtaining a file defense rule;

obtaining a container label set for the file defense rule; and

determining an association relationship between the file defense rule and the container label;

where a device driver is configured to: detect a file access request initiated by an application program, and determine a container identifier of a target container corresponding to the application program, where the container identifier is used to determine an associated container label, the container label is used to determine at least one corresponding file defense rule by searching for the association relationship, and the at least one file defense rule is used to determine whether to perform a defense action on the file access request.

[0011] According to a sixth aspect, an embodiment of the present application provides a computing device, including a processing component and a storage component;

where the storage component stores one or more computer instructions, and the one or more computer instructions are invoked and executed by the processing component, to implement the file protection method according to the first aspect, or the file protection method according to the second aspect, or the file protection method according to the third aspect, or the rule configuration method according to the fourth aspect, or the file protection method according to the fifth aspect.

[0012] According to a seventh aspect, an embodiment of the present application provides a computer storage medium, storing a computer program, where the computer program, when executed by a computer, implements the file protection method according to the first aspect, or the file protection method according to the second aspect, or the file protection method according to the third aspect, or the rule configuration method according to the fourth aspect, or the file protection method according to the fifth aspect.

[0013] In the embodiments of the present application, a device driver in a kernel detects a file access request initiated by an application program, may determine a container identifier of a target container from which the application program comes, so that a corresponding container label may be determined according to the container identifier, may search for at least one corresponding file defense rule according to the container label, and may determine, according to the at least one file defense rule, whether to perform a defense action on the file access request. A defense range to which the file defense rule is applicable is set based on the container label, so that protection in different defense ranges is implemented, and a container file is effectively protected. In addition, the defense action is implemented in combination

with the device driver in the kernel, thereby improving protection stability and protection efficiency.

[0014] These aspects or other aspects of the present application are more concise and easily understandable in descriptions of the following embodiments.

BRIEF DESCRIPTION OF DRAWINGS

[0015] To describe the technical solutions in the embodiments of the present application or the prior art more clearly, the following briefly introduces the accompanying drawings required for describing the embodiments or the prior art. Apparently, the accompanying drawings in the following descriptions show some embodiments of the present application, and a person of ordinary skill in the art may still derive other drawings from these accompanying drawings without creative efforts.

[0016] FIG. 1 shows a diagram of an architecture of a system to which a technical solution is applied according to an embodiment of the present application.

[0017] FIG. 2 shows a flowchart of an embodiment of a file protection method according to the present application.

[0018] FIG. 3 shows a flowchart of an embodiment of another file protection method according to the present application.

[0019] FIG. 4 shows a flowchart of an embodiment of another file protection method according to the present application.

[0020] FIG. 5 shows a flowchart of an embodiment of a rule configuration method according to the present application.

[0021] FIG. 6 shows a schematic diagram of display of a rule configuration interface in an actual application according to an embodiment of the present application.

[0022] FIG. 7 shows a flowchart of an embodiment of another file protection method according to the present application.

[0023] FIG. 8 shows a schematic structural diagram of an embodiment of a file protection apparatus according to the present application.

[0024] FIG. 9 shows a schematic structural diagram of an embodiment of another file protection apparatus according to the present application.

[0025] FIG. 10 shows a schematic structural diagram of an embodiment of another file protection apparatus according to the present application.

[0026] FIG. 11 shows a schematic structural diagram of an embodiment of a rule configuration apparatus according to the present application.

[0027] FIG. 12 shows a schematic structural diagram of an embodiment of another file protection apparatus according to the present application.

[0028] FIG. 13 shows a schematic structural diagram of an embodiment of a computing device according to the present application.

DESCRIPTION OF EMBODIMENTS

[0029] To make a person skilled in the art understand solutions in the present application better, the following clearly and completely describes the technical solutions in the embodiments of the present application with reference to the accompanying drawings in the embodiments of the present application.

[0030] Some procedures described in the specification, the claims, and the accompanying drawings of the present application include a plurality of operations occurred in a particular order. However, it should be clearly learned that these operations may not be performed in the order in which these operations are occurred herein or may be performed concurrently. Sequence numbers, such as 101 and 102, of the operations are merely used to distinguish different operations, and the sequence numbers themselves do not represent any execution order. In addition, these procedures may include more or fewer operations, and these operations may be performed in order or performed concurrently. It should be noted that descriptions of "first", "second", and the like herein are used to distinguish different messages, devices, modules, and the like, do not indicate a sequence, and do not limit that "first" and "second" to be of different types.

[0031] The technical solutions in the embodiments of the present application are applicable to a container file protection scenario, for example, a container file protection scenario in a cloud computing scenario. A container may be a container running in a physical

machine or running in a virtual machine built in the physical machine, which is not limited in the present application.

[0032] For ease of understanding the technical solutions in the present application, the following firstly make corresponding explanations for technical terms possibly related in the present application.

[0033] Container technology: the container technology is used to effectively divide resources of a single operating system into isolated groups, to better balance conflicting resource use requirements among the isolated groups. This technology is the container technology. Containers created based on the container technology are similar to virtual machines, but they have relaxed isolation properties, to share the operating system between application programs. The container in the embodiments of the present application may be created in a host machine, and the host machine may be a physical host or a virtual machine such as a cloud server, which is not limited in the present application.

[0034] Kubernetes: the Kubernetes, K8s for short, is an open-source container management platform, and may be configured to manage containers on a plurality of machines in a cloud computing platform.

[0035] Virtual machine (VM): the virtual machine is a complete computer system that has a complete hardware system function and runs in a completely isolated environment and that is simulated by using software. In a cloud computing scenario, a virtual machine may be, for example, an elastic cloud server.

[0036] Container image: a relationship between the container image and a container is similar to that between a class and an instance in an object-oriented program design. An image is static, and a container may be an entity for a container image at runtime.

[0037] Host machine: the host machine may be a physical machine or a virtual machine. The host machine may run the container and another application program. An application program may alternatively run inside the container. A difference between the application program and the program running in the host machine lies in that, unless the container is specially configured, generally, the application program can access only specific resources or data inside the container.

[0038] Kernel: the kernel is a core of an operating system, is a first-layer software extension based on hardware, provides a most basic function of the operating system, is a basis for the operation of the operating system, and is responsible for managing a system process, memory, driver program, file, and a network system, thereby determining system performance and stability.

[0039] Application layer (user mode): a modern operating system provides two running states: a user mode and a kernel mode. Generally, all application programs (processes) run in the user mode. In this case, the process cannot directly access a hardware resource or system data. If needed, the process enters the kernel through a system call, and continues to obtain a related resource or data through the kernel. Finally, the process returns to the user mode and continues to run.

[0040] System call (syscall): the system call is a programmatic way in which a computer program requests a service from the operating system (kernel) in which the program is executed. The service may include a hardware-related service (for example, accessing a hard disk drive or accessing a camera of a device), creation and execution of a new process, and communication with an integrated kernel service (for example, process scheduling). The system call provides a basic interface between the process and the operating system. In the embodiments of the present application, all file access operations of the application program need to be implemented by invoking the system call.

[0041] Application program: the application program is an execution program created in the container, and is used to implement different execution operations. Each application program corresponds to one process. In a cloud computing scenario, a user may run different application programs and the like in one or more containers. Certainly, the application program may alternatively be created and run in the host machine running outside the container.

[0042] In actual application, the technical solutions in the embodiments of the present application may be applied to a cloud computing scenario. Cloud computing is one of the fastest trends in computer technology development, and involves providing a hosting service by using a network. In a cloud computing environment, computing and storage resources are provided to end users as services. The end users may request the provided services for processing. Processing capabilities of the services are typically limited by configured resources.

[0043] It should be understood that although detailed descriptions of cloud computing are included in the present application, implementation of the teaching described herein is not limited to be in the cloud computing environment. On the contrary, the embodiments of the present application can be implemented with reference to any other type of computing environment known now or developed later.

[0044] Cloud computing is a service delivery model, aiming to implement on-demand network access to a shared pool of configurable computing resources (for example, networks, network bandwidths, servers, processing, memories, storage, application programs, virtual machines, and services), where the resources can be rapidly provisioned and released with a minimal management effort or through interaction with a service provider.

[0045] The cloud server is a cloud computing service provided by a cloud computing provider, and the cloud computing provider may gather resources by using a multi-tenant model to form a resource pool to provide services for a plurality of tenants. The user herein is a tenant. A resource of the cloud computing provider may be purchased by means of funds, to set up a cloud server meeting requirements of the user. The user may register a user account with the cloud computing provider, and the cloud computing provider distinguishes different users by using user accounts. One or more containers and the like may be further created in the cloud server.

[0046] In a process of implementing the present application, the inventor finds that, although a current container management tool such as K8s (Kubernetes, an open-source container management platform) provides a resource isolation function between containers, so that an application program in one container cannot access a file in another container, only file protection between containers can be implemented in this file protection manner. In actual application, for example, a specific file in a container does not want to be maliciously modified by some application programs. In this case, the application programs are not excluded from a same container. In this case, a file cannot be effectively protected in a resource isolation manner provided by the K8s, and a potential security risk still exists.

[0047] To improve file protection effectiveness and avoid a potential security risk, the inventor proposes the technical solutions in the embodiments of the present application through a series of studies. In the embodiments of the present application, a device driver in a kernel detects a file access request initiated by an application program, may determine a container identifier of a target container from which the application program comes, so that a

corresponding container label may be determined according to the container identifier, may search for at least one corresponding file defense rule according to the container label, and may determine, according to the at least one file defense rule, whether to perform a defense action on the file access request. A defense range to which the file defense rule is applicable is set based on the container label, so that protection in different defense ranges is implemented, and a container file is effectively protected. In addition, the defense action is implemented in combination with the device driver in the kernel, thereby improving protection stability and protection efficiency.

[0048] The technical solutions in the embodiments of the present application are clearly and completely described in the following with reference to the accompanying drawings in the embodiments of the present application. Apparently, the embodiments to be described are merely a part rather than all of the embodiments of the present application. All other embodiments obtained by a person skilled in the art based on the embodiments of the present application without creative efforts shall fall within the protection scope of the present application.

[0049] FIG. 1 shows a diagram of an architecture of a system to which a technical solution may be applied according to an embodiment of the present application. The system architecture may include a server end 101. An operating system and at least one container 102 are deployed in the server end 101, and the operating system may include a kernel 103 and an application layer 104. In addition, the operating system includes different system types, for example, currently popular Windows, Linux, and MacOS, which is not limited in the present application. The present application is applicable to any system that can support a container environment, for example, Linux.

[0050] One or more application programs (processes) 105 may run in each container 102.

[0051] A defense program 106 may be deployed in the application layer 104, and after the defense program 106 is started, a corresponding device driver 107 may run in the kernel 103. The device driver 107 may detect all system calls related to file access in the kernel, to determine whether to perform a defense action on a file access request initiated by any application program from any container.

[0052] In this embodiment of the present application, the server end 101 may be a physical host, certainly may be a cloud server that provides a basic cloud computing service

such as a cloud service, a cloud database, cloud computing, a cloud function, cloud storage, a network service, cloud communication, a middleware service, a domain name service, a security service, a content delivery network (CDN), a big data, and an artificial intelligence platform, or may be an intelligent cloud computing server or an intelligent cloud host with an artificial intelligence technology.

[0053] It should be noted that, in the embodiments of the present application, use of user data may be involved. In actual application, when requirements of applicable laws and regulations of the country where the present application is practiced (for example, with explicit consent of the user, or effective notification to the user) are met, user-specific personal data can be used in the solution described herein within the scope permitted by applicable laws and regulations.

[0054] The implementation details of the technical solutions in the embodiments of the present application are described in detail in the following.

[0055] FIG. 2 is a flowchart of an embodiment of a file protection method according to an embodiment of the present application. A technical solution of this embodiment is performed by a device driver in a kernel. The method may include the following steps.

[0056] 201: detect a file access request initiated by an application program.

[0057] The application program may invoke a system call in the kernel to initiate a file access request for a target file. The device driver may detect a system call related to a file access operation in the kernel, to determine whether there is the file access request initiated by the application program.

[0058] The file access request may include file information of the target file. The file information may include, for example, a file path, a file name, a file type, and/or an operation type of the target file. Where the operation type is used to indicate different file access operations, and may include, for example, a read operation, a write operation, and a modification operation.

[0059] 202: determine a container identifier of a target container to which the application program belongs.

[0060] The corresponding container identifier may be found according to process information of the application program. The process information may include, for example, a process identifier.

[0061] Optionally, a search may be conducted for the container identifier corresponding to the application program from kernel data. The kernel data is data that is in the system kernel and that can be accessed by the application program after entering a system kernel mode. A current (an information field in the kernel data) of the kernel data includes process related information of the application program, and the corresponding container identifier may be found from the current of the kernel data. An application program that is in a container and that initiates a file access request enters the kernel through the system call. In this case, the device driver may obtain data of the application program (process) by accessing the current of the kernel data, and further determine, according to the data, a container identifier to which the application program belongs.

[0062] In an optional manner, the container identifier may be represented by using a container ID (Identity document, identity authentication identifier).

[0063] In another optional manner, since the container ID is typically a 64-byte string, when information searching is performed based on the container ID, matching overheads are relatively large, and the information searching efficiency is relatively slow. Therefore, optionally, the container identifier may be represented by using a namespace identifier. The namespace identifier is typically 32-bits unsigned integer data. This can reduce matching overheads and improve the information searching efficiency.

[0064] The namespace identifier may be, for example, a network namespace (net-ns) identifier, a process identifier-namespace (pid-ns), or a file namespace (mnt-ns, mount-namespace) identifier. Different containers typically use independent and different file namespaces, and the file namespace identifier may uniquely identify a file system. Therefore, the file namespace identifier may be used as the container identifier, to distinguish different containers.

[0065] The above target container may be a container to which the application program belongs, which is named as a "target container" for ease of description.

[0066] 203: search for a container label associated with the container identifier.

[0067] 204: search for at least one corresponding file defense rule according to the container label.

[0068] In this embodiment of the present application, container labels corresponding to different file defense rules may be preconfigured, and a container label is used to define a file defense range of a file defense rule. The container label may be used to identify a cluster, an image, an application type, or the like corresponding to a container, to limit the file defense range. Therefore, optionally, the container label may include at least one of an application label used to identify an application type, a cluster label used to identify a cluster, or an image label used to identify a container image.

[0069] The application label is used to identify an application type corresponding to a container, so that file protection may be performed from an application dimension. The cluster label is used to identify a cluster corresponding to the container, and may be implemented by using a cluster ID, so that file protection may be performed from a cluster dimension. The image identifier is used to indicate an image corresponding to the container, and may be implemented by using an image ID, so that file protection may be performed from an image dimension.

[0070] Since a defense program may detect and learn of a startup operation of any container, the defense program may determine a cluster label of a cluster to which the container belongs, an image label of an image to which the container belongs, a corresponding namespace identifier of the container, a container identifier of the container, or the like, and may generate container related information according to the cluster label, the image label, the namespace identifier, the container identifier, or the like, and deliver the container related information to the device driver. Therefore, the device driver may search, according to the container related information delivered by the defense program, for the container label associated with the container identifier. The container related information may include a correspondence between the container identifier and the container label. When the container identifier is the namespace identifier, the container related information includes a correspondence between the namespace identifier and the container label. The defense program may associate the container label with the namespace identifier, to generate the container related information and deliver the container related information to the device driver. The container related information may exist in a form of a data table such as a hash table or a red black tree, which is not limited in the present application.

[0071] Optionally, a corresponding file defense rule may be set for one or more labels.

[0072] When the container label associated with the container identifier includes a plurality of labels, a search may be conducted for a corresponding file defense rule for one or more of the labels, so that at least one file defense rule may be obtained.

[0073] In actual application, the defense program may be first installed in a host machine. After the defense program is installed and started, the defense program may install the device driver in the kernel, to detect a system call related to a file operation. The defense program may send container labels that correspond to different file defense rules and that are preconfigured to the device driver.

[0074] The defense program may detect starting and stopping of a container in the host machine. For a started container, the defense program may obtain a container identifier and a container label, and send data of the container identifier and the container label to the device driver. For a stopped container, the defense program may still obtain a container identifier of the stopped container, and may send the container identifier to the device driver to delete an associated label corresponding to the container identifier.

[0075] Therefore, the device driver may search, according to the container identifier and container label that are associated and that are delivered by the defense program, for the container label corresponding to the container identifier, and search, according to container labels that correspond to different file defense rules and that are delivered by the defense program, for file defense rules corresponding to the container labels.

[0076] 205: determine, according to the at least one file defense rule, whether to perform a defense action on the file access request.

[0077] Optionally, it may be determined whether file information in the file access request matches the at least one file defense rule, and it is determined, according to a determining result, whether to perform the defense action on the file access request. If the determining result is that the file information matches any file defense rule, the defense action may be performed on the file access request. If the determining result is that the file information does not match the at least one file defense rule, the file access request may be allowed, so that a file can be protected in a file defense range (for example, the application dimension, the cluster dimension, or the image dimension) defined by the container label, thereby improving file access security. In this way, an application program belonging to a

specific application type, a specific cluster, or a specific image cannot implement a file access operation on a target file.

[0078] The file defense rule may include a to-be-defended file path, file name, file type, or the like. Therefore, it may be determined, according to the file information in the file access request and according to the to-be-defended file path, file name, file type, or the like in the file defense rule, whether the file access request hits the file defense rule. If the file access request hits the file defense rule, the file information matches the file defense rule, and it may be determined to perform the defense action on the file access request. If the file access request does not hit the file defense rule, the file access request may be allowed, and the kernel is invoked to continue to perform a corresponding file access operation or the like.

[0079] In addition, the file defense rule may further include a process whitelist, used to set information about an application program that is allowed to access a file corresponding to the set to-be-defended file path, file name, file type, or the like. Optionally, the process whitelist may be empty. Therefore, it may be first determined, according to the file defense rule, whether the application program is located in the process whitelist. If the application program is located in the process whitelist, the file access request may be directly allowed. If the application program is not located in the process whitelist, it may be determined again, with reference to the file information in the file access request, and the to-be-defended file path, file name, file type, or the like in the file defense rule, whether the file access request hits the file defense rule.

[0080] In addition, the file defense rule may further include an action type, used to set a defense action that is performed when the file defense rule is hit. The action type may include, for example, outputting alarm information or performing an interception operation. Therefore, when it is determined to perform the defense action on the file access request, a corresponding defense action may be performed according to the action type in the file defense rule. If the file information hits a plurality of file defense rules, a plurality of defense actions may be performed according to action types respectively corresponding to the plurality of file defense rules.

[0081] In addition, priorities may alternatively be set for the plurality of labels included in the container label. A file defense rule corresponding to each label may be sequentially searched in descending order of the priorities. If there is a corresponding file defense rule, the search may be stopped. The file defense rule is a final corresponding file defense rule. When

the application label and the cluster label are found according to the container identifier, the application label has a highest priority, and a search may be preferentially conducted for a file defense rule corresponding to the application label. If there is the file defense rule corresponding to the application label, the operation of step 205 is performed by using the file defense rule. If there is no file defense rule corresponding to the application label, a search may be conducted for a corresponding file defense rule according to the cluster label.

[0082] In this embodiment, a device driver in a kernel detects a file access request initiated by an application program, to determine a container identifier of a target container from which the application program comes and determine a corresponding container label according to the container identifier, and search for a corresponding file defense rule according to the container label; and the device driver may determine, according to the file defense rule, whether to perform a defense action on the file access request. A defense range to which the file defense rule is applicable is set based on the container label, so that protection in different defense ranges is implemented, and a container file is effectively protected. In addition, the defense action is implemented in combination with the device driver in the kernel, thereby improving protection stability and protection efficiency.

[0083] In some embodiments, it can be learned from the foregoing descriptions that the container label includes at least one of the application label used to identify the application type, the cluster label used to identify the cluster, or the image label used to identify the container image.

[0084] The method may further include: obtaining an association relationship that is between a file defense rule and a container label and that is delivered by the defense program.

[0085] The searching for a corresponding file defense rule according to the container label may include: searching for the association relationship according to the container label, to determine a corresponding file defense rule.

[0086] In other words, the defense program may obtain user configuration information, and deliver the association relationship between the file defense rule and the container label in the user configuration information to the device driver, so that the device driver may search for the corresponding file defense rule according to the association relationship.

[0087] FIG. 3 is a flowchart of an embodiment of another file protection method according to an embodiment of the present application. A technical solution of this

embodiment may be performed by a device driver in a kernel. The method may include the following several steps.

[0088] 301: detect a file access request initiated by an application program, and determine file information.

[0089] The application program may invoke a system call in the kernel to initiate a file access request for a target file. The device driver may detect a system call related to a file access operation in the kernel, to determine whether there is the file access request initiated by the application program

[0090] The file access request may include the file information, so that the file information can be obtained. The file information may include, for example, a file path, a file name, a file type, and/or an operation type. Where the operation type is used to indicate different file access operations, and may include, for example, a read operation, a write operation, and a modification operation.

[0091] 302: determine a container identifier of a target container corresponding to the application program.

[0092] For a specific execution operation of step 302, refer to the descriptions in step 202. Details are not described herein again.

[0093] 303: send the container identifier and the file information to a defense program, so that the defense program searches for a container label associated with the container identifier, searches for at least one corresponding file defense rule according to the container label, and generates indication information according to a matching result between the file information and the at least one file defense rule.

[0094] 304: determine, according to the indication information sent by the defense program, whether to perform a defense action on the file access request.

[0095] A difference between this embodiment and the embodiment shown in FIG. 2 lies in that the device driver delivers the container identifier and the file information to the defense program, and the defense program searches for the container label, finds the file defense rule according to the container label, and performs a matching operation.

[0096] In an optional manner, the indication information may include the matching result. The device driver may determine, according to the matching result, whether to perform the

defense action on the file access request. When the matching result indicates that the file information matches the file defense rule, the defense action is performed on the file access request. When the matching result indicates that the file information does not match the file defense rule, the file access request may be allowed.

[0097] In another optional manner, the defense program may generate an execution instruction according to the matching result, and the indication information may include the execution instruction generated according to the matching result. The execution instruction may indicate, to the device driver, whether to perform the defense action, so that the device driver does not need to perform a determining operation again.

[0098] The defense program may pre-store an association relationship between the file defense rule and the container label, so that a search may be conducted for the association relationship according to the container label to determine the at least one corresponding file defense rule.

[0099] When detecting a startup operation of any container, the defense program may obtain container related information, to search, according to the container related information, for the container label associated with the container identifier.

[0100] The file defense rule may include a to-be-defended file path, file name, file type, or the like. Therefore, the defense program may determine, according to the file information in the file access request and according to the to-be-defended file path, file name, file type, or the like in the file defense rule, whether the file access request hits the file defense rule. If the file access request hits the file defense rule, the file information matches the file defense rule. If the file access request does not hit the file defense rule, it may be considered that the file information does not match the file defense rule.

[0101] In addition, the file defense rule may further include a process whitelist, used to set information about an application program that is allowed to access a file corresponding to the set to-be-defended file path, file name, file type, or the like. Optionally, the process whitelist may be empty. Therefore, the defense program first determines whether the application program is located in the process whitelist. If the application program is located in the process whitelist, it may be considered that the file information does not match the file defense rule. If the application program is not located in the process whitelist, it may be determined, with reference to the file information, and the to-be-defended file path, file name,

file type, or the like in the file defense rule, whether the file access request hits the file defense rule.

[0102] In addition, the file defense rule may further include an action type, used to set a defense action that is performed when the file defense rule is hit. The action type may include, for example, outputting alarm information or performing an interception operation. Therefore, when determining that the file information matches the at least one file defense rule, the defense program may further send the action type to the device driver, and the indication information may include the action type, to indicate the device driver to execute a corresponding defense action according to the action type.

[0103] In this embodiment, a device driver in a kernel detects a file access request initiated by an application program, to determine a container identifier of a target container from which the application program comes, so that a defense program determines a corresponding container label according to the container identifier, and may search for at least one corresponding file defense rule according to the container label; and the device driver may determine, according to the at least one file defense rule, whether to perform a defense action on the file access request. A defense range to which the file defense rule is applicable is set based on the container label, so that protection in different defense ranges is implemented, and a container file is effectively protected. In addition, the defense action is implemented in combination with the device driver in the kernel, thereby improving protection stability and protection efficiency.

[0104] FIG. 4 is a flowchart of an embodiment of another file protection method according to an embodiment of the present application. The technical solution in this embodiment is performed by a defense program. The defense program runs in an application layer of an operating system. In actual application, the defense program is first installed in a host machine. After the defense program is installed and started, the defense program may install a device driver in a kernel, to detect a system call related to a file operation. The method may include the following several steps.

[0105] 401: obtain a container identifier and file information that are sent by a device driver.

[0106] The file information is determined by the device driver by detecting a file access request initiated by an application program, and the container identifier is obtained through

searching by the application program. The device driver may determine the container identifier from kernel data.

[0107] The container identifier may be implemented by using a namespace identifier, for example, a file namespace identifier identifying a file system.

[0108] 402: search for a container label associated with the container identifier.

[0109] 403: search for at least one corresponding file defense rule according to the container label.

[0110] The defense program may detect starting and stopping of a container in the host machine. For a started container, the defense program may obtain a container identifier and a container label. Since the defense program may detect and learn of a startup operation of any container, the defense program may determine a cluster label of a cluster to which the container belongs, an image label of an image to which the container belongs, a corresponding namespace identifier of the container, a container identifier of the container itself, or the like, and may generate container related information according to the cluster label, the image label, the namespace identifier, the container identifier, or the like. Therefore, a search may be conducted for the container label associated with the container identifier according to the container related information. The container related information may include a correspondence between the container identifier and the container label. When the container identifier is the namespace identifier, the container related information includes the correspondence between the namespace identifier and the container label. The defense program may associate the container label with the namespace identifier, to generate the container related information and deliver the container related information to the device driver. The container related information may exist in a form of a data table such as a hash table or a red black tree, which is not limited in the present application.

[0111] 404: generate indication information according to a matching result between the file information and the at least one file defense rule.

[0112] 405: send the indication information to the device driver, where the indication information is used to indicate whether to perform a defense action on the file access request.

[0113] In an optional manner, the indication information may include the matching result. The device driver may determine, according to the matching result, whether to perform the

defense action on the file access request. When the matching result indicates that the file information matches the file defense rule, the defense action is performed on the file access request. When the matching result indicates that the file information does not match the file defense rule, the file access request may be allowed.

[0114] In another optional manner, the defense program may generate an execution instruction according to the matching result, and the indication information may include the execution instruction generated according to the matching result. The execution instruction may indicate, to the device driver, whether to perform the defense action, so that the device driver does not need to perform a determining operation again.

[0115] The defense program may pre-store an association relationship between the file defense rule and the container label, so that the association relationship may be searched according to the container label to determine the at least one corresponding file defense rule.

[0116] When detecting a startup operation of any container, the defense program may obtain container related information, to search, according to the container related information, for the container label associated with the container identifier.

[0117] In this embodiment, a device driver in a kernel detects a file access request initiated by an application program, to determine a container identifier of a target container from which the application program comes, so that a defense program determines a corresponding container label according to the container identifier, and may search for a corresponding file defense rule according to the container label; and the device driver may determine, according to the file defense rule, whether to perform a defense action on the file access request. A defense range to which the file defense rule is applicable is set based on the container label, so that protection in different defense ranges is implemented, and a container file is effectively protected. In addition, the defense action of the file access request is implemented in combination with the device driver in the kernel, thereby improving protection stability and protection efficiency.

[0118] In addition, a file defense rule may be configured and obtained by a user. Referring to FIG. 5, FIG. 5 is a flowchart of an embodiment of a rule configuration method according to an embodiment of the present application. The technical solution in this embodiment may be performed by a client. The method may include the following several steps.

[0119] 501: provide rule configuration prompt information on a rule configuration interface.

[0120] The rule configuration prompt information is used to prompt a user to perform a configuration operation and the like.

[0121] 502: in response to a user configuration operation, determine a file defense rule configured by a user and a container label set for the file defense rule.

[0122] The rule configuration prompt information may include, for example, rule setting prompt information and label setting prompt information.

[0123] The rule setting prompt information may include file information setting prompt information, whitelist setting prompt information, action setting prompt information, and the like.

[0124] The file information setting prompt information is used to prompt a user to configure file information. The whitelist setting prompt information is used to prompt the user to configure a process whitelist. The action setting prompt information is used to prompt the user to configure an action type and the like. Therefore, the file defense rule may be obtained with reference to the file information and the process whitelist and/or the action type.

[0125] The label setting prompt information may be used to prompt the user to set a container label.

[0126] 503: send the file defense rule and the container label to a defense program, so that the defense program determines an association relationship between the file defense rule and the container label.

[0127] Optionally, the file defense rule and the container label may be sent to the defense program through a control end.

[0128] Where a device driver is configured to: detect a file access request initiated by an application program, and determine a container identifier of a target container corresponding to the application program. The container identifier is used to determine an associated container label, and the container label is used to determine at least one corresponding file defense rule by searching for the association relationship. The at least one file defense rule is used to determine whether to perform a defense action on the file access request.

[0129] In an optional manner, the defense program may determine the association relationship between the file defense rule and the container label, and deliver the file defense rule and the association relationship to the device driver. The device driver is configured to: detect a file access request initiated by an application program, search for a container identifier of a target container corresponding to the application program, search for the association relationship according to a container label associated with the container identifier, to determine at least one corresponding file defense rule, and determine, according to the at least one file defense rule, whether to perform a defense action on the file access request.

[0130] Additionally, the defense program may detect starting and stopping of a container in a host machine. For a started container, the defense program may obtain a container identifier and a container label, and send data of the container identifier and the container label to the device driver. For a stopped container, the defense program may still obtain a container identifier of the container, and may send the container identifier to the device driver to delete an associated label corresponding to the container identifier.

[0131] Therefore, the device driver may search, according to the container identifier and container label that are associated and that are delivered by the defense program, for the container label corresponding to the container identifier, and search, according to container labels that correspond to different file defense rules and that are delivered by the defense program, for file defense rules corresponding to the container labels.

[0132] In an optional manner, the defense program may determine the association relationship between the file defense rule and the container label, obtain a container identifier and file information that are sent by the device driver, search for a container label associated with the container identifier, search for at least one corresponding file defense rule according to the container label, generate indication information according to a matching result between the file information and the at least one file defense rule, and send the indication information to the device driver.

[0133] The device driver may detect a file access request initiated by an application program, determine file information, determine a container identifier of a target container corresponding to the application program, send the container identifier and the file information to the defense program, and determine, according to indication information sent by the defense program, whether to perform a defense action on the file access request.

[0134] For ease of understanding, FIG. 6 shows a schematic interface diagram of a rule configuration interface. The rule configuration interface shown in FIG. 6 may include rule setting prompt information and label setting prompt information 601. The rule setting prompt information may include file information setting prompt information 602, whitelist setting prompt information 603, action setting prompt information 604, and the like. The rule configuration interface may further include confirmation prompt information, which is used by a user to perform a confirmation operation on a configured file defense rule and a container label, so that sending of the file defense rule and the container label to a defense program is triggered.

[0135] The label setting prompt information may include at least one piece of label type selection prompt information and label input prompt information corresponding to each piece of label selection prompt information. The at least one piece of label type selection prompt information may be first displayed. In response to a label type selected by the user, label input prompt information corresponding to the selected label type may be further displayed.

[0136] In addition, in some embodiments, the method may further include:

performing a corresponding processing operation in response to an update operation on the file defense rule.

[0137] The update operation may include, for example, deletion, modification, or the like, where the modification operation may include, for example, a modification operation on file information, a process whitelist, an action type, or a corresponding container label.

[0138] FIG. 7 is a flowchart of an embodiment of another file protection method according to an embodiment of the present application. A technical solution in this embodiment is performed by a defense program. The method may include the following several steps.

[0139] 701: obtain a file defense rule.

[0140] 702: obtain a container label set for the file defense rule.

[0141] The file defense rule and the container label may be determined by a client in response to a user configuration operation, and the client may send the file defense rule and the container label to the defense program through a control end.

[0142] 703: determine an association relationship between the file defense rule and the container label.

[0143] The defense program may establish the association relationship between the file defense rule and the container label. Therefore, a device driver detects a file access request initiated by an application program, and determines a container identifier of a target container corresponding to the application program. The container identifier is used to determine an associated container label, the container label is used to determine at least one corresponding file defense rule by searching for the association relationship, and the at least one file defense rule is used to determine whether to perform a defense action on the file access request.

[0144] In an optional manner, the defense program may determine the association relationship between the file defense rule and the container label, and deliver the file defense rule and the association relationship to the device driver. The device driver is configured to: detect a file access request initiated by an application program, search for a container identifier of a target container corresponding to the application program, search for the association relationship according to a container label associated with the container identifier, to determine at least one corresponding file defense rule, and determine, according to the at least one file defense rule, whether to perform a defense action on the file access request.

[0145] In an optional manner, the defense program may determine the association relationship between the file defense rule and the container label, obtain a container identifier and file information that are sent by the device driver, search for a container label associated with the container identifier, search for at least one corresponding file defense rule according to the container label, generate indication information according to a matching result between the file information and the at least one file defense rule, and send the indication information to the device driver.

[0146] The device driver may detect a file access request initiated by an application program, determine file information, determine a container identifier of a target container corresponding to the application program, send the container identifier and the file information to the defense program, and determine, according to indication information sent by the defense program, whether to perform a defense action on the file access request.

[0147] In some embodiments, the defense program may further deliver the file defense rule and the association relationship to the device driver.

[0148] Therefore, the device driver may detect a file access request initiated by an application program, determine a container identifier of a target container corresponding to the application program, search for the association relationship according to a container label associated with the container identifier, to determine a corresponding file defense rule, and determine, according to the file defense rule, whether to perform a defense action on the file access request.

[0149] In some embodiments, the method may further include:

detecting a startup operation of any container, and sending container related information of the container to the device driver, where the device driver searches, according to the container related information, for the container label associated with the container identifier.

[0150] FIG. 8 is a schematic structural diagram of an embodiment of a file protection apparatus according to an embodiment of the present application. The apparatus may include:

a first detection module 801, configured to detect a file access request initiated by an application program;

a first determining module 802, configured to determine a container identifier of a target container to which the application program belongs;

a first searching module 803, configured to search for a container label associated with the container identifier;

a second searching module 804, configured to search for at least one corresponding file defense rule according to the container label; and

a first execution module 805, configured to determine, according to the at least one file defense rule, whether to perform a defense action on the file access request.

[0151] In some embodiments, the first searching module is specifically configured to search, according to container related information delivered by a defense program, for the container label associated with the container identifier. The defense program is configured to send obtained corresponding container related information to a device driver when detecting a startup operation of any container.

[0152] In some embodiments, the container label includes at least one of an application label used to identify an application type, a cluster label used to identify a cluster, or an image label used to identify a container image.

[0153] The apparatus may further include:

a first obtaining module, configured to obtain an association relationship that is between a file defense rule and a container label and that is delivered by the defense program.

[0154] The second searching model may be specifically configured to search for the association relationship according to the container label, to determine the at least one corresponding file defense rule.

[0155] In some embodiments, the first determining module may be specifically configured to search, from kernel data, for a namespace identifier corresponding to the application program, where the namespace identifier is used to identify the target container to which the application program belongs.

[0156] In some embodiments, the namespace identifier includes a file namespace identifier.

[0157] In some embodiments, the first execution module may be specifically configured to determine whether file information in the file access request matches the at least one file defense rule, and determine, according to a determining result, whether to perform the defense action on the file access request.

[0158] The file protection apparatus described in FIG. 8 may perform the file protection method described in the embodiment shown in FIG. 2. The implementation principles and technical effects thereof are not described herein again. For the file protection apparatus in the foregoing embodiment, a specific manner in which each module and unit perform operations is already described in detail in the embodiment related to the method. Details are not described herein again.

[0159] FIG. 9 is a schematic structural diagram of an embodiment of another file protection apparatus according to an embodiment of the present application. The apparatus may include:

a second detection module 901, configured to detect a file access request initiated by an application program and determine file information;

a second determining module 902, configured to determine a container identifier of a target container corresponding to the application program;

a first sending module 903, configured to send the container identifier and the file information to a defense program, so that the defense program searches for a container label associated with the container identifier, searches for at least one corresponding file defense rule according to the container label, and generates indication information according to a matching result between the file information and the at least one file defense rule; and

a second execution module 904, configured to determine, according to the indication information sent by the defense program, whether to perform a defense action on the file access request.

[0160] The file protection apparatus described in FIG. 9 may perform the file protection method described in the embodiment shown in FIG. 3. The implementation principles and technical effects thereof are not described herein again. For the file protection apparatus in the foregoing embodiment, a specific manner in which each module and unit perform operations is already described in detail in the embodiment related to the method. Details are not described herein again.

[0161] FIG. 10 is a schematic structural diagram of an embodiment of another file protection apparatus according to an embodiment of the present application. The apparatus may include:

a second obtaining module 1001, configured to obtain a container identifier and file information that are sent by a device driver, where the file information is determined by the device driver by detecting a file access request initiated by an application program, and the container identifier is obtained through searching based on the application program;

a third searching module 1002, configured to search for a container label associated with the container identifier;

a fourth searching module 1003, configured to search for at least one corresponding file defense rule according to the container label;

a matching module 1004, configured to generate indication information according to a matching result between the file information and the at least one file defense rule; and

a second sending module 1005, configured to send the indication information to the device driver, where the indication information is used to indicate whether to perform a defense action on the file access request.

[0162] The file protection apparatus described in FIG. 10 may perform the file protection method described in the embodiment shown in FIG. 4. The implementation principles and technical effects thereof are not described herein again. For the file protection apparatus in the foregoing embodiment, a specific manner in which each module and unit perform operations is already described in detail in the embodiment related to the method. Details are not described herein again.

[0163] FIG. 11 is a schematic structural diagram of an embodiment of a rule configuration apparatus according to an embodiment of the present application. The apparatus may include:

a first display module 1101, configured to provide rule configuration prompt information on a rule configuration interface;

a first processing module 1102, configured to in response to a user configuration operation, determine a file defense rule configured by a user and a container label set for the file defense rule; and

a third sending module 1103, configured to send the file defense rule and the container label to a defense program, so that the defense program determines an association relationship between the file defense rule and the container label;

where a device driver is configured to: detect a file access request initiated by an application program, and determine a container identifier of a target container corresponding to the application program, where the container identifier is used to determine an associated container label, the container label is used to determine at least one corresponding file defense rule by searching for the association relationship, and the at least one file defense rule is used to determine whether to perform a defense action on the file access request.

[0164] The rule configuration apparatus described in FIG. 11 may perform the rule configuration method described in the embodiment shown in FIG. 5. The implementation principles and technical effects thereof are not described herein again. For the rule configuration apparatus in the foregoing embodiment, a specific manner in which each

module and unit perform operations is already described in detail in the embodiment related to the method. Details are not described herein again.

[0165] FIG. 12 is a schematic structural diagram of an embodiment of another file protection apparatus according to an embodiment of the present application. The apparatus may include:

a third obtaining module 1201, configured to: obtain a file defense rule, and obtain a container label set for the file defense rule; and

a relationship determining module 1202, configured to determine an association relationship between the file defense rule and the container label;

where a device driver is configured to: detect a file access request initiated by an application program, and determine a container identifier of a target container corresponding to the application program, where the container identifier is used to determine an associated container label, the container label is used to determine at least one corresponding file defense rule by searching for the association relationship, and the at least one file defense rule is used to determine whether to perform a defense action on the file access request.

[0166] In some embodiments, the apparatus may further include:

a fourth sending module, configured to deliver the file defense rule and the association relationship to the device driver, so that the device driver detects a file access request initiated by an application program, determines a container identifier of a target container corresponding to the application program, and configured to search for the association relationship according to a container label associated with the container identifier, to determine at least one corresponding file defense rule, and determine, according to the at least one file defense rule, whether to perform a defense action on the file access request.

[0167] In some embodiments, the apparatus may further include:

a fifth sending module, configured to: detect a startup operation of any container, and send container related information of the container to the device driver, where the device driver searches, according to the container related information, for the container label associated with the container identifier.

[0168] The file protection apparatus described in FIG. 12 may perform the file protection method described in the embodiment shown in FIG. 7. The implementation principles and

technical effects thereof are not described herein again. For the file protection apparatus in the foregoing embodiment, a specific manner in which each module and unit perform operations is already described in detail in the embodiment related to the method. Details are not described herein again.

[0169] An embodiment of the present application also provides a computing device. As shown in FIG. 13, the computing device may include a storage component 1301 and a processing component 1302.

[0170] The storage component 1301 stores one or more computer instructions, and the one or more computer instructions are invoked and executed by the processing component 1302, to implement the file protection method according to the embodiment shown in FIG. 2, or the file protection method according to the embodiment shown in FIG. 3, or the file protection method according to the embodiment shown in FIG. 4, or the rule configuration method according to the embodiment shown in FIG. 5, or the file protection method according to embodiment shown in FIG. 7.

[0171] The computing device may run one or more containers.

[0172] Certainly, the computing device may necessarily further include other components, for example, an input/output interface, a display component, and a communication component.

[0173] When the processing component in the computing device is configured to implement the rule configuration method according to the embodiment shown in FIG. 5, the computing device may further include the display component to perform a corresponding display operation.

[0174] The input/output interface is an interface provided between the processing component and a peripheral interface module, and the peripheral interface module may be an output device, an input device, or the like. The communication component is configured to facilitate communication in a wired or wireless manner between the computing device and another device.

[0175] The processing component may include one or more processors to execute computer instructions, so as to complete all or some steps of the foregoing methods. Certainly, the processing component may alternatively be implemented by using one or more

application specific integrated circuits (ASICs), digital signal processors (DSPs), digital signal processing devices (DSPDs), programmable logic devices (PLDs), field programmable gate arrays (FPGAs), controllers, microcontrollers, microprocessors, or other electronic elements, to execute the foregoing methods.

[0176] The storage component is configured to store various types of data to support operations on a terminal. The storage component may be implemented by using any type of volatile or non-volatile storage devices or a combination thereof, for example, a static random access memory (SRAM), an electrically erasable programmable read-only memory (EEPROM), an erasable programmable read-only memory (EPROM), a programmable read-only memory (PROM), a read-only memory (ROM), a magnetic memory, a flash memory, a disk, or an optical disc.

[0177] The display component may be an electroluminescent (EL) element, a liquid crystal display or a micro-display having a similar structure, or a retinal direct display or a similar laser scanning display.

[0178] It should be noted that, when the computing device implements the file protection method according to the embodiment shown in FIG. 2, or the file protection method according to the embodiment shown in FIG. 3, or the file protection method according to the embodiment shown in FIG. 4, or the file protection method according to embodiment shown in FIG. 7, the computing device may be a physical device, an elastic computing host provided by a cloud computing platform, or the like. The computing device may be implemented as a distributed cluster including a plurality of servers or terminal devices, or may be implemented as a single server or a single terminal device. The processing component may alternatively be a running virtual machine, so that the virtual machine implements the file protection method according to the embodiment shown in FIG. 2, or the file protection method according to the embodiment shown in FIG. 3, or the file protection method according to the embodiment shown in FIG. 4, or the file protection method according to embodiment shown in FIG. 7.

[0179] When the computing device implement the rule configuration method shown in FIG. 5, the computing device may be implemented as an electronic device. The electronic device may be a device that is used by a user and that has functions such as computing, network access, and communication needed by the user, for example, may be a mobile phone, a tablet computer, a personal computer, or a wearable device.

[0180] An embodiment of the present application further provides a computer-readable storage medium, storing a computer program. The computer program, when executed by a computer, may implement the file protection method according to the embodiment shown in FIG. 2, or the file protection method according to the embodiment shown in FIG. 3, or the file protection method according to the embodiment shown in FIG. 4, or the rule configuration method according to the embodiment shown in FIG. 5, or the file protection method according to embodiment shown in FIG. 7. The computer-readable medium may be included in the electronic device described in the foregoing embodiment, or may exist alone and is not disposed in the electronic device.

[0181] An embodiment of the present application further provides a computer program product, including a computer program carried in a computer-readable storage medium. The computer program, when executed by a computer, may implement the file protection method according to the embodiment shown in FIG. 2, or the file protection method according to the embodiment shown in FIG. 3, or the file protection method according to the embodiment shown in FIG. 4, or the rule configuration method according to the embodiment shown in FIG. 5, or the file protection method according to embodiment shown in FIG. 7. In such an embodiment, the computer program may be downloaded and installed from a network, and/or installed from a removable medium. When the computer program is executed by a processor, various functions defined in the system of the present application are executed.

[0182] A person skilled in the art may clearly understand that, for the purpose of convenient and brief description, he or she may refer to a corresponding process in the above method embodiments for a detailed working process of the system, apparatus, and unit described above. Details are not described herein again.

[0183] The foregoing described apparatus embodiments are merely examples. The units described as separate parts may or may not be physically separate, and the parts displayed as units may or may not be physical units, that is, the parts may be located in one position, or may be distributed in a plurality of network units. Some or all the modules may be selected according to actual needs to achieve the objectives of the solutions of the embodiments. A person of ordinary skill in the art may understand and implement the embodiments of the present application without creative efforts.

[0184] Based on the foregoing descriptions of the implementations, a person skilled in the art may clearly understand that each implementation may be implemented by software in

addition to a necessary general hardware platform or by hardware. Based on such an understanding, the foregoing technical solutions essentially or the part thereof contributing to the prior art may be implemented in a form of a software product. The software product may be stored in a computer readable storage medium, such as a ROM/RAM, a hard disk, or an optical disc, and include several instructions for instructing a computer device (which may be a personal computer, a server, a network device, or the like) to perform the methods described in the embodiments or some parts of the embodiments.

[0185] Finally, it should be noted that the foregoing embodiments are merely intended for describing the technical solutions of the present application, but not for limiting the present application. Although the present application is described in detail with reference to the foregoing embodiments, a person of ordinary skill in the art should understand that he or she can still modify the technical solutions described in the foregoing embodiments or equivalently replace some of the technical features thereof; and such modifications or replacements do not cause the essence of the corresponding technical solutions to depart from the spirit and scope of the technical solutions of the embodiments of the present application.