Research on Docker Role Access Control Mechanism Based on DRBAC

Dapeng Lang and Haochen Jiang* and Wei Ding and Yu Bai
*Haerbin Engineering University, Heilongjiang Province, China
E-mail: 1974457552@qq.com Tel: +86-13206755967

Abstract— **With the rapid development of virtualization technology, security issues have also become increasingly prominent. All kinds of virtualization platforms use access control mechanisms to prevent illegal or legal entities from illegally accessing unauthorized resources. This paper discusses the access control mechanism adopted in Docker container, analyzes its shortcomings, and proposes a DRBAC based Docker role access control mechanism. This method is based on the role allocation method, dynamically expands the existing model, and adds a role access control mechanism to the Docker container. Through the distributed trust management and access control mechanism of DRBAC, we can solve the hidden danger brought by the main body because of delegated authority to other entities autonomously. This scheme can effectively control the access management and delegate problem of the main body when accessing Docker resources, and improve the access security of Docker container to host resources.**

Key word：DRBAC, Docker container, access control, virtualization technology

I.BACKGROUND

With the rapid development and popularization of cloud technology, the high performance and portability of container technology are becoming more and more popular; which are widely concerned and widely promoted [1]. But the security of access between the container and the host is also greatly challenged. Access control technology is utilized to make sure that unauthorized information system resources are not accessed illegally by authorized or unauthorized entities. As a basic important safety technology, it is widely used in all kinds of information system and operating systems [2] .

At present, various access control techniques are used in various virtual technologies and containers. The existing Docker access permission techniques contains the autonomic access control mechanism provided by some Linux operating systems, and the Namespace mechanism and Cgroups mechanism in the Linux kernel. The latter two technologies are used to isolate the resources between the Docker containers and the Linux host, Namespaces mechanism and Cgroups mechanism are also the basis for access control mechanisms between the Docker container and the Linux host. However, not all resources in the Linux operating systems are under a namespace. Processes of the Docker container can access unrelated kernel resources, such as /sys, /proc, etc. in the root directory. Once the Docker container process needs to be able to access these files, it may cause the kernel resources leak, and more seriously. The Docker container user may gain control permissions for the entire host machine

This article analyzes the isolation mechanism of Docker container for access control in Linux host. Meanwhile we present the arbitrary distribution of users' permission in autonomous access control which leads to the problem of lower security. At last, it presents the problem of poor flexibility in mandatory access control. To solve the defects existing in the control of independent access control and compulsory access control, this article proposes a Docker role access control mechanism based on DRBAC to improve the safety of containers

II.RESEARCH STATUS

The access control model consists of three types. Discretionary Access Control, Mandatory Access Control, and Role-Based Access Control. Each of these models has advantages and disadvantages, among them, the discretionary access control model has the characteristics of flexibility and autonomy, and users can grant some of their own access permission or full access permission to the other user. This model will cause the system to fail to control the information flow and to determine the user's scope of permission. Access control is employed in multilevel security systems, following strict hierarchy rules. Therefore, the flexibility of the model is not just enough. It is both difficult and important to apply in a dynamic environment. The main idea of Role-Based Access Control is to add roles between users and permissions in the traditional access control model, by controlling the grant and revoke of the user's permission through roles, it can effectively reduce the complexity of the authorization management work caused by a mount of users of the system.

At present, in the cloud-based virtual environment, the main technology is the access control technology based on attributes, as mentioned in article [3], Cloud computing access control addresses architecture, mechanism, and model issues. In attribute control, among the attribute control, many types are classified, such as access control technology based on attribute encryption and attribute access control optimization

This work was supported by the free exploration program of Harbin Engineering University (No. HEUCF180607).

technology based on trust evaluation [4]. Docker is based on the autonomous access control technology[5] which comes with Linux. At the same time it uses the role access control technology[6] in the Selinux system, and proposes to add a dynamic role access control mechanism in the Docker access control mechanism.

In summary, in the virtual platform and container, the access control technology requires (1) flexibility, (2) controllability, and (3) scalability. These qualities ensure that authorized users securely access authorized resources.

III.BASIC PRINCIPLES AND IMPLEMENTATION PROCESS

A. Docker Access Mechanism.

The Docker container access control mechanism is based on the Namespaces mechanism and Cgroups mechanism. It uses the two mechanisms in the Linux kernel to achieve isolation and quota of containers. The Docker container utilizes the Namespaces namespace mechanism to dismiss processes, networks, messages, file systems, UTS, and Linux hosts in the Docker container. The Docker uses Cgroups mechanism to limit and measure the system resources, so as to control the system resources that the Docker container process can use.

a. Namespaces mechanism

The Linux operating system uses the Linux Namespaces namespace mechanism to separate the Linux system resources. After Docker uses the LinuxNamespaces namespace mechanism, the Linux system resources become local resources, belonging to a fixed Namespaces. Two system resources under the same namespace are invisible to each other, and the system resources under different namespaces are invisible to each other. For example, from the perspective of operation system, PIDNamespace will have multiple PID processes, but they belong to different namespaces, and they will not conflict with each other.

b. Cgroups mechanism

The full name of Cgroups is the Control Groups control group. The Linux operating system distributes the physical resources of the Linux system by using the control group technology. The control group mechanism is the underlying support of the container technology in the Linux system, which provides the basis for the virtualization of the container. The Linux kernel uses the Cgroups control group mechanism to restrict, record and isolate the Linux operating system resources. The Cgroups mechanism is made up of different Cgroups subsystems, and any process and system resources are grouped under different subsystems. There is a Cgroups virtual file system in the Linux system. The virtual file system provides Linux users with interfaces to manage and set up each Cgroups subsystem. Users can set up and manage the Cgroups control group corresponding to the CGroup subsystem they need to use.

B. Docker Role Assignment Mechanism

By default, Docker-initiated containers are strictly limited to allow only some of the kernel's features. This is due to the Linux kernel's ability to provide fine-grained access control. Servers will run a bunch of processes that require privilege privileges, including ssh, cron, syslogd, hardware management tool modules (such as load modules), network configuration tools etc. Almost all privileged processes are managed by a support system other than the container. For example: ssh access is managed by the ssh service on the host; cron is usually performed as a user process, permissions are given to applications that use it; logs are managed by Docker or third-party services. So the kernel only needs to allocate some functions to the Docker container without having to be a real root permission, so that it can satisfy its authority requirements.

C. Basic Concepts Of DRBAC

DRBAC model is a role-based distributed access control model proposed by Freudenthal et al. The model uses PKI to identify and verify the identity and authentication of the trusted operating entity, thus realizing the access control problem of resources in dynamic alliance environment across multiple management domains. The DRBAC model provides an extended distributed trust management and access control mechanism. This model has the following three characteristics: third party commission, value attribute, and certificate subscription.

DRBAC combines the advantages of RBAC and trust management systems, which are systems that manage both flexible and decentralized, extensible implementations. DRBAC describes a controlled behavior based on roles. The role is defined within the trust domain of an entity, and it can delegate the role to other roles within different trust domains. DRBAC uses PKI to identify all entities associated with trust-sensitive operations and acknowledges assignment certificates. Mapping from role to authorization namespace avoids the need to identify additional policy roots.

D. DRBAC Allocation Syntax

In the DRBAC basic model. The syntax of the delegation is: [Subject→Object] Issuer Where Object is a role, the Issuer is an entity, and the Subject is a role or entity. The meaning of syntax is the signing certificate declares that the role Object grants access to the Subject

There are three types of commissions in the DRBAC model.

a. object delegation: [Subject→A.a]A, the implication is that publisher A grants A.a to Subject, the A.a is a role that A definition of A's own namespace.

b. assignment delegation:[Subject→A.a]B, the implication is that B assigns delegate authority of role A.a to Subject. After that, the Subject can delegate the role A.a to another Subject.

c. third-party delegation:[Subject→A.a]B, the implication is that publisher B delegates role A to Subject, where an and B are different entities

E. PKI technology

PKI technology is based on public key technology and relies on digital certificates to bind the information of entities on the network to their respective public keys. By automatically managing keys and certificates, a secure and reliable network environment is established for users, and users can use data encryption and digital signature techniques to verify the identity of users on the Internet in a variety of applications. In order to ensure the authenticity, integrity, confidentiality and non repudiation of information transmitted on the Internet. PKI is the only technology that enables user authentication and ensures data security on the Internet so far.

A typical PKI system mainly includes the following parts: Certification: CA, Registration Authority: RA, certificate issuing body and PKI application.The PKI functional structure showed in Fig.1.

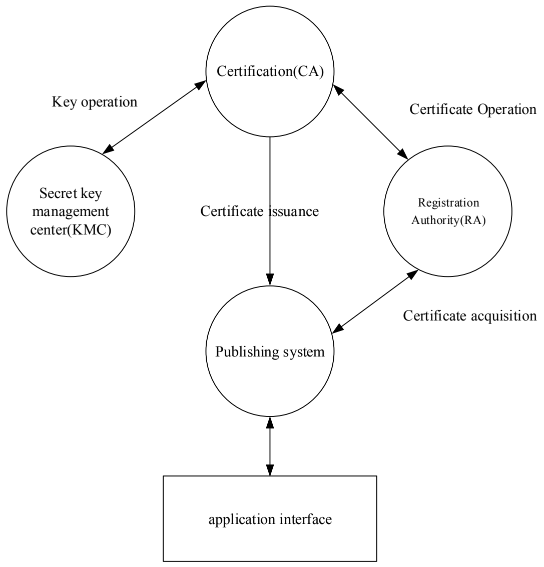

Fig.1 PKI functional structure

F. DRBAC Role Assignment Process

a. Authentication

DRBAC combines the identity authentication of PKI/PMI architecture to verify the legality of user identity. When the user logs in to the application system, he/she first performs user authentication, and if successful, he/she obtains the permission attribute certificate and obtains it from the LDAP server.

b. Access Control

According to the permit property certificate of role permission information, information about role of inheriting, the change of role mapping information and role constraints of information constitute the current user authorization information, to determine whether the user has the right to access to the system resources

c. Configuration Management

Providing a convenient and friendly graphical user interface for system administrators. System administrators can perform user management, role management, rights management, role/rights management, rule development, and define inheritance and constraint policies.

d. Permission Attribute Certificate Authentication

When an access resource request is issued after the user logs in to the system, authentication server to authenticate user's rights attribute information, and according to the access control policy, a determination is made to determine the subsequent operation sequence.

IV. RAC FRAMEWORK BASED ON DRBAC

A．DRBAC Distributive Grammar

First, an object delegation is performed, and it’s used to assign part of the privilege of OS to a user administrator, so that the user administrator becomes the entity that assigns roles. In this way of distributing authority, even if attacked, only part of OS's permission is promised, not the whole root's permission.

Eg：[Subject1→ OS. ssh] OS

Using an assignment of delegate, the privileges of a user administrator with a part of the OS authority are allocated to other users, so that the user administrator can assign appropriate permissions according to the specific needs of the user.

Eg：[Subject2→ Guser1. ssh’] subject1

Proxy entrustment: Suppose there are two heterogeneous application systems, set as domain A and domain B. There is a role R1 in the A domain, and a role R2 in the B domain. If R1 wants to obtain the rights of role R2 in the B domain to perform R2 operations, but R1 is not recognized in B domain, then role R1 can request R2 to complete the corresponding function and return the result to R1. R1 and R2 are actually a relationship of proxy delegate. The role mapping mechanism between domains can be implemented by proxy delegation mechanism. Then achieve inter domain role cross domain access, it is suitable for distributed systems.

This role inheritance method not only supports single inheritance, but also supports multiple inheritance. A role can inherit multiple roles at the same time, thereby inheriting the rights of multiple roles. This can create new roles conveniently on the basis of the original roles. The characteristics of multiple inheritance make DRBAC have the following two advantages: First, multiple complex roles can be constructed by multiple inheritance of several simple roles at the bottom (with fewer permissions). Second, multiple inheritance features provide unified user / role assignment and role inheritance relationship to application systems.

The most important extension of the distributed DRBAC model to the traditional RBAC is to cross the mapping between the roles of the system. A good role mapping transformation mechanism between heterogeneous systems is the key technology of DRBAC model.

The role mapping transformation between this domain and outfield can be achieved by building two domain based role inheritance trees. Then, the corresponding roles are established to achieve the inter domain role mapping transformation and cross domain access.

The role mapping between domains is showd in Fig.1.In the inter domain role mapping model, the mapping of code 1 represents the transitive mappings from A1R1 to A0R0, as follows: A1Rl->A0R0. Through this mapping, roles on A1R1 and above can be converted to role A0R0. In the role inheritance hierarchy of H1. Because the role Admin1 >A1 can be mapped to A0R0 by mapping A1R1 −>A0R0 with label 1. The mapping of label 2 is a non transitive mapping from B1R1 to C0R0. Remember to do: B1R1-> (NT) C0R0. NT indicates that the mapping is a non transitive map, which means that B1R1 can be converted to C0R0. Although the permissions of roles Admin1R1 and A1R1 are higher than B1R1, the mapping of the label 2 is a non - transitive mapping, so these two roles can not be converted to C0R0 by mapping B1R1 ->C0R0.

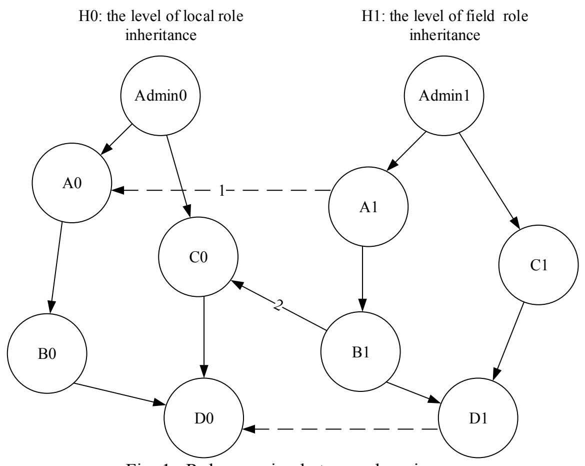
H0: the level of local role inheritance
H1: the level of field role inheritance
Admin0
Admin1
A0
1
A1
C0
C1
2
B0
B1
D0
D1

Fig. 1 Role mapping between domains.

B. Specific Allocation Method.

a. Configure System Files

The different file systems used by the Linux system are not accessed by the device identifier, instead they are connected into a single tree structure that represents a file system with a single unified entity. The kernel must support the ext2 file system before operating on the kernel. We set CONFIG_EXT2_FS to Y. The same configuration supports file properties to set CONFIG_EXT2_FS_XATTR to Y, configuration support ACL, configuration support ACL, configuration support for on-chip execution, etc.

b. Compile The Kernel.

After compiling the kernel file, we obtain a strategy which contains the data. Then we use the *make* command to compile this strategy into a binary policy module (under Fedora 23 environment). Its security context is as followed: System_u: object_r: semanage_ exec_t as is showed in Figure 2.

c. Authorize the Role

In order for the user administrator to have the ability to grant roles to different users within Docker, administrator needs to have the root user and the sysadm_t domain. However, root and sysadm_t domain types cannot be directly assigned to all user administrators. If the authority is granted directly, the user administrator has the right to modify the policy source code file directly, so that it has the authority to grant all the roles. At this point, each role is authorized to create a separate program named [role name] assign for the user, and the program file attributes are set to have higher authority.

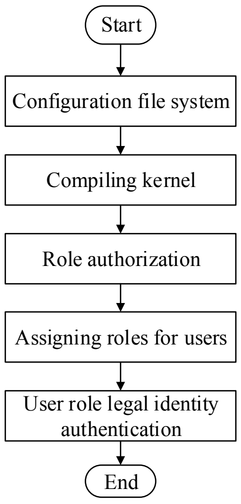
Start
Configuration file system
Compiling kernel
Role authorization
Assigning roles for users
User role legal identity authentication
End

Fig. 2 allocation flow chart.

Although the user administrator role dsm_r has sysadm_t, it only has this type when executing the [role name] assign program.And after the process is finished running, it no longer has this type. Because the security context of the user who has this role logs in is dsm_u: dsm_r: dsm_t, and dsm_t itself has very limited permissions, it can only run [role name] assign a type of file. After executing the file, SHELL automatically fork a child process to perform the operation, the child process will automatically inherit the parent process's security context. Because of the automatic domain conversion rules and the allowed domain transformation rules, the domain types of the child processes are converted to sysadm_t. After this permission is raised, the child process can assign roles to other users (the user name is passed through the command parameter). The revocation of user roles is done in the same way.

d. Assigns Roles to Users

Two ordinary users A_u and B_u are created. At the same time we specify the roles that these two users can be related to. User B_u can only use the B_r role, while A_u can only use A_r. A prefix is specified for the user's home directory type. For user A, the administrator specify the user name and extend the home directory to A_home_dir_t, and use

A_home_t for files in that directory. At this point, the allocation of users and roles is completed. The description is showed in Figure 3.

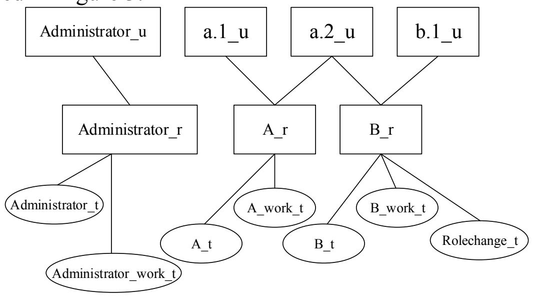

Fig. 3 role assignment diagram

e. Role User's Legal Identity Authentication

When using proxy delegate, Docker users enter the access control module to authenticated user identity. By querying the LDAP directory server to verify whether it is a legitimate user. If it is a legitimate system user, get the corresponding attribute certificate for user binding. Otherwise, the user will be denied access. After the identity is confirmed, the control right is handed over to the access controller. The first step of access control is to distinguish whether it is the local tenant or the remaining tenants for local tenants, they get permission from attribute certificates and go into the access control decision modular. Otherwise, they should transform their roles into local ones first, then determine the access control. The access decision module, based on the current user's permission information and the pre-defined access control strategy, draws a decision conclusion. If the user has the authority, it will be accessed; otherwise, the user will be denied access. The implementation process of DRBAC is as followed in Figure 4.

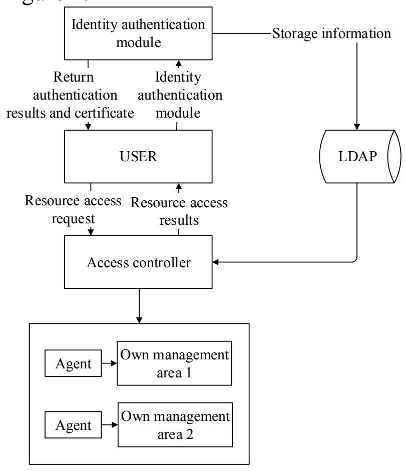

Fig. 4 implementation process of DRBAC

V. CONCLUSION

This paper introduced the advantages and disadvantages of autonomic access control and mandatory access control in traditional access control. The access control mechanism in Docker was based on the traditional access control problem, a framework of Docker container role access control mechanism based on DRBAC was proposed. The framework uses DRBAC access control technology to manage flexible and to implement the system dispersedly features. To solve Discretionary Access Control permission to address the subject is too large, it leads to the problem of information leakage, so that it can be applied in a more complex hierarchical system. At the same time, due to strict compliance with hierarchical rules, the flexibility of mandatory access control is not enough, so it is both difficult and important to apply in dynamic environment, and to achieve cross-domain access between the roles of the domain, applicable to multi-tenant in Docker. The implementation shows that our mechanism works well to make up other security access mechanisms.

REFERENCES

[1] Lu Tao, Chen Jie, Shi Jun. Research of Docker Security [J/OL]. computer technology and development, 2018 (06): 1-6 [2018-05-03].

[2] Yang Wenlin, Tan Xi, Guo Junting, Wang Shuo. The Vulnerability Analysis and Security Enhancement of Docker [J]. information security and technology, 2016,7 (04): 21-23+55.

[3] Wang Yu Ding, Yang Jiahai, Xu Cong, Ling Xiao, Yang Yang. Survey on Access Control Technologies for Cloud Computing [J]. software journal, 2015,26 (05): 1129-1150.

[4] Liu An. Cloud computing access control technology [J]. electronic technology and software engineering, 2018 (07): 183.

[5] Li Pingping, Chen Lijun. Research on Docker access control mechanism based on LSM [J]. information technology, 2016 (11): 134-138+142.

[6] Jianghua. Research on SELinux Role一based Access Control Technology [D]. Huazhong University of Science and Technology, 2009.