Research on Container-Oriented Isolation Control Technology

Baohua Zhao1,2,3, Zhihao Wang1,2, Ningyu An1,2 and Chunhui Ren1,2*

1 Global Energy Interconnection Research Institute Co.Ltd, Beijing, 102209, China
2 Artificial Intelligence on Electric Power System State Grid Corporation Joint Laboratory (GEIRI), Beijing, 102209, China
3 Beijing University of Technology, Beijing, 100124, China
*Corresponding author’s e-mail: renchunhui@geiri.sgcc.com.cn

Abstract. Container technology has a series of advantages such as low physical resource consumption, fast startup speed, high concurrency, and can run in a variety of environments. It is widely used in scenarios such as big data and cloud computing. Container technology has certain advantages in performance, but there are some shortcomings in security. The container technology shares the kernel with the host, and its security mainly depends on the host. Once the attacker breaks through the host's defense, he can easily access the files deployed in the container, steal or tamper with the file data, and cause losses to users and users. In response to the above problems, this paper proposes a container-oriented isolation control technology, which realizes further isolation of files inside the container by adding domain names to programs and files. If the program domain name matches the file part, the files in the current container cannot be accessed, and the security of the files in the container can be effectively ensured after the host is compromised.

1. Introduction

Container technology has a series of advantages such as low physical resource consumption, fast startup speed, and high concurrency. When applied to cloud computing, big data and other scenarios, it can effectively improve resource management efficiency[1]. With the advent of the big data era, the demand for resources continues to increase, and the scenario of container technology has become more and more extensive[2], and the development of container technology has become increasingly important[3]. This article takes the mainstream Docker container as an example, analyzes the security problems of the current container technology, and proposes corresponding solutions.

In recent years, a large number of cloud platforms deploying Docker containers have problems with data loss or data leakage[4], exposing the lack of container security. At present, the security of Docker container technology essentially depends on the security of container isolation provided by the namespace mechanism of the host, the security of control capabilities provided by the control group mechanism, and the security of operation permissions provided by the capability mechanism[5]. Once the attacker breaks through the host's defense, he can easily access the files inside the container, steal or tamper with the file data, and cause losses to users and users[6].

In summary, this article proposes a container-oriented isolation control technology, through a security control strategy, based on the "namespace isolation security" technology, a scope is
established to implement mandatory access control to the internal files of the container. This access control solution has the following advantages:

- It acts directly on the system kernel and is relatively safe and reliable.
- It has the same effect on the host file and the internal files of the container, improving the overall security of the container and the host;
- It supports functions such as internal migration of server clusters and group management, which has a small impact on container performance.

2. Related technology

Container technology can package and isolate applications and their dependencies into a single object, which has a series of advantages such as less physical resource consumption, fast startup speed, high concurrency and so on[7-8]. Docker is an advanced container engine, which encapsulates and abstracts the underlying technologies such as container namespace and cgroups, and provides an efficient, convenient and lightweight container solution[9]. Through its own advantages, it has been favored by many companies and has become the current mainstream container technology.

Container technology can solve the problems of high resource consumption and low work efficiency in industries such as cloud computing and big data. Yang P. et al. believe that the cloud platform can optimize the computing performance of the system and the performance of file writing by virtue of the lightweight virtualization advantages of Docker containers[10]. Wu Z.X. et al. pointed out that Docker will become the core technology of cloud computing[11], so the development of container technology is very critical. At present, because of the incomplete resource isolation of the container technology, the security is more dependent on the security of the host, there is a greater security risk, and related scholars have carried out research on this[12].

Han S.H. et al.[13] proposed to isolate the access control architecture of the container image from the container engine and dynamically identify the files in the container image. Only users authorized by the container platform interface have the right to access the files, protecting the container image. Li P.P. et al.[14] proposed a solution to implement Docker container process access control to the Linux host from the kernel level, effectively preventing the Docker container process from accessing unisolated kernel resources in the Linux host. Wang J. et al.[15] proposed a Docker-based container trusted reinforcement scheme, which uses trusted computing technology, trust chain technology, hook technology, etc. to construct a trust chain from hardware to container internal program and files to achieve all-round Measurement and fine-grained monitoring of Docker.

The above solution lacks protection of files inside the container, and has deficiencies such as limiting user operations and high resource consumption. In order to solve the above problems, this paper proposes a container access technology based on the Linux namespace isolation security mechanism, which implements the mandatory access control to the internal files of the container through the namespace or scope, and supports the internal migration and group management of the container technology server cluster.

3. Container-oriented isolation control technology solution

This solution is based on a naming control mechanism to achieve further isolation between processes and files. When the container is running, add a domain name composed of the container's "image name: container ID" to related programs and files in the system. The access control strategy is realized by judging whether the domain names of programs and files are consistent. The scheme is implemented based on docker container engine. The domain name of host is defined as system. The system block diagram is shown in Figure 1.
In figure 1, the administrator configures the security access strategy on the management platform, and implements file security access through the control engine in the client kernel. program0 and file0 are the programs and files of the host machine itself; program1 and file1 are programs and files corresponding to the Docker1 container; program2 and file2 are the programs and files corresponding to the Docker2 container. The program and file connected by solid line have the same domain name, which conforms to the isolation control policy. The program can access the file, which is judged by the control engine in the kernel. The virtual line indicates that the connected program and file have different domain names, which does not comply with the access control policy. The file and program are isolated, and the program with different domain names cannot access the file.

The process of adding domain names to programs and files is combined with container whitelist scanning. The client calls the “apphash” instruction to start the whitelist maintenance program. The management platform periodically controls the client to issue the application layer instruction "Docker images" to obtain the existing Docker image and image id of the host, and automatically generate the corresponding image list. After the mirror list is generated, the system automatically starts to scan the internal files of the mirror in the mirror list. After the scan, the client invokes the application layer command "Docker save -o path+image name.tar image name/image ID", adds a temporary file storage directory /usr/ses/images under the installation directory /usr/ses, and exports The image of is stored here by default. The client calls the apphash command at the application layer to scan and release the directory where the tar package is located, and at the same time add domain names for programs and files. After the whitelist scan is completed, clean up the directory. The schematic diagram of the whitelist scan file is shown in figure 2.
When the program of the host or container initiates the request to access the file, it judges whether the docker protection module of the isolation control is in the running state. When the docker protection module is closed, the system directly rejects the access request and closes the program. When the docker protection module is turned on, it judges whether the domain name of the program and the domain name of the file comply with the access relationship policy in the access control module, and the program with the same domain name and file is allowed to execute through authorization. An unauthorized program or program that does not comply with the access control policy is judged to be accessed illegally and reported to the management center. The schematic diagram of program access control is shown in figure 3.
container image name and program id and other information, the above information is saved in the security file, and the subsequent program is started to search for the corresponding access control policy subject and object for use. The process and file domain names are obtained through the white list scanning process. After obtaining all the information required by the access control policy, whether the program can access the file is judged by comparing the program and file domain names according to the access control policy. Process access file control process diagram is shown in figure 4.

![Diagram](image_url)

Figure 4. program access file control process diagram.

This solution realizes the isolation of files inside the container by assigning domain names to processes and files, and further improves the security of files inside the container. In the scheme, the access control policy is formulated and issued by the management platform, which supports the internal migration and group management of server cluster, so as to avoid the limitation of container technology function and affect the application in various scenarios.

4. Conclusion

With the characteristics of lightweight, low deployment cost and management boundary, container technology is widely used in cloud computing, big data and other scenarios with high resource demand. In this paper, the security of container technology relies too much on the host, and once the host is compromised, the internal files of the container are easy to leak and lose. This paper proposes a container-oriented isolation control technology. The process ID of the container is used as the domain name of the file and process, and the file and process in the container are isolated from the process on the host through the namespace mechanism. When the host is at risk, the security of the files in the container is effectively improved. This technology is issued by the management center and directly acts on the system kernel. It has high reliability. It can also support the cross-platform migration of containers and their security access policies, and has little impact on container performance.
Acknowledgments
This work is supported by the science and technology project of State Grid Corporation of China: “Research on Key Technologies of Fundamental Security Protection and Control for IoT Smart Terminals” (Grand No. 5700-201958464A-0-0-00).

References
[1] Shu, A., Peng, X., Zhao, W.Y. (2017) Cloud computing resource adaptive management method based on container technology. Computer Science, 44(07):120-127.
[2] Yang, Y. (2021) Discussion on the overall protection technology of cloud platform information security. China Management Informationization, 24(02):200-201.
[3] Yuan, Z.L. (2017) Research on Key Technologies of Container Cloud Computing Platform. Nanjing University.
[4] Nikolai, J., Wang, Y. (2016) A System for Detecting Malicious Insider Data Theft in IaaS Cloud Environments. In:2016 IEEE Global Communications Conference (GLOBECOM). Washington, DC. pp. 1-6.
[5] Chen, W., Tu, J.L. (2020) Analysis and Research on Container Security. Communications Technology, 53(12):3072-3077.
[6] Su, J. (2020) Research on Container Security Control Technology. Network Security Technology & Application, 11:21-22.
[7] Xu, Y.Q., Huang, H., Jin, Z. (2021) Research on the Application of Container Technology in Scientific Computing. Computer Science, 48(01):319-325.
[8] Shantharama, P., Thyagaturu, A.S., Yatavelli, A., Lalwaney, P., Reisslein, M., Tkachuk, G., Pullin, E.J. (2020) Hardware Acceleration for Container Migration on Resource-Constrained Platforms. IEEE Access, 8:175070-172085.
[9] Chen, Q.J., Chen, C.X., Zhang, Y. (2015) Docker technology implementation analysis. Information and Communications Technologies, 9(02):37-40.
[10] Yang, P., Ma, Z.C., Peng, B., Yao, J.G. (2017) Performance Research of OpenStack Cloud Platform Integrated with Docker Container. Computer Engineering, 43(08):26-31.
[11] Wu, Z.X. (2017) The development and trend of cloud computing virtualization technology. Journal of Computer Applications, 37(04):915-923.
[12] Kwon, S., Lee, J. (2020) DIVDS: Docker Image Vulnerability Diagnostic System. IEEE Access, 8:42666-42673.
[13] Han, S.H., Lee, H.K., Lee, S.T., Kim, S.J., Jang, W.J. (2020) Container Image Access Control Architecture to Protect Applications. IEEE Access, 8:162012-162021.
[14] Li, P.P., Chen, L.J (2016) Research on Docker Access Control Mechanism Based on LSM. Information Technology, 11:134-138+142.
[15] Wang, J., Hu, W., Zhang, Y.W., Chen, T., Yu, P., Zhao, B., Zhang, H.G. (2017) Docker-based trusted container. Journal of Wuhan University(Natural Science Edition), 63(02):102-108.