Research Article

Design Scheme of a Docker Container File Isolation against Computer Virus Spreading

Hanji Ju,1 Jie Wang,1 Enguo Zhu,2 Xiaoli Zhang,1 and Feng Zheng3

1Metrology Center, State Grid JiBei Electric Power Co Ltd, Beijing 102208, China
2China Electric Power Research Institute, Beijing 100192, China
3School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Feng Zheng; zhengfeng@bupt.edu.cn

Received 9 December 2021; Revised 13 January 2022; Accepted 26 February 2022; Published 28 March 2022

Copyright © 2022 Hanji Ju et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the spread of computers and Internet of Things, the destructive power of viruses gradually becomes stronger, which threaten the daily life of people. In order to effectively restrain the virus spreading and reduce economic losses, humans need to hold the law and control of the virus distribution. Docker has obtained significant attention as a lightweight virtualization container technology. Compared to traditional virtualization technologies, docker possesses the advantages of fast deployment, low resource consumption, and high migration capability. Given that the differences between various virtualization technologies and combining the characteristics in electric power systems, this paper designs a scheme of the file isolation system based on docker containers. The isolation scheme could reduce the highly dependence on the underlying operating system in the current application deployment of electric power systems and facilitate the security access to the file system between containers.

1. Introduction

The spreading dynamics based on computer virus becomes a new interdisciplinary subject. Through dynamic system modeling, we can understand the law of virus distribution and provide an effective virus containment strategy. The different kinds of computer viruses possess different functions, which can cause huge losses to national security and property security. Some computer viruses can damage software and hardware resources, including invading the financial systems and stealing finances, tamper data, and misleading administrators. Moreover, some computer viruses can implant medical chips and harm human health. Therefore, more and more researchers propose schemes to prevent the spread of the virus.

A virus attack on the power system would potentially bring down the power infrastructure and thus affect daily life. In Ukraine, an attack on the power system led to widespread power outages in 2015 [1]. It is therefore important to monitor the power grid [2], to remove the risk of potential attack. While detecting, appropriate risk assessment strategies should be developed to quantify the risk thresholds [3] so that possible risks can be addressed promptly. At the same time, setting different security levels for nodes in the grid system can also contain the spread of viruses to a certain extent [4]. Among current research solutions, most of the attacks on the grid are detected using more advanced artificial intelligence algorithms [5]. However, the detection of the system does not allow for the complete isolation of viruses between systems, and it is possible that viruses can only be detected when they are already a serious threat to the operational security of the power system.

With the development of information technology and the construction of national information projects, more application systems and related software are gradually not satisfied with the current deployment method [6]. In order to improve the utilization of each hardware resource, as well as to get rid of the situation of over reliance on the grassroots operating system in the power system, the national grid
information system is vigorously promoting the solution of deploying traditional application systems to the cloud through virtualization technology. The cloud-based deployment solution also enables the formation of different security levels between different power systems, which will effectively stop the spread of viruses between power systems by forming different security differentials.

Traditional application deployment based on power systems does not show too many shortcomings in a single application deployment and a small number of task management. However, when the amount of management tasks needs to dynamically change and migrate tasks, this deployment scheme will show its shortcomings of the bulky deployment. The problems can be effectively resolved by deploying applications in docker containers [7], which is the mainstream solution in the current virtualization direction [8]. However, docker container-based application deployment solutions have their own potential problems. For example, docker container technology as an operating system-level virtualization technology itself will appear between the container and the host with incomplete isolation, cumbersome security reinforcement, and other problems so that there may be isolation between the container escape, the container and the host paralysis, and the potential danger of core data leakage [9]. The construction of electric power systems, as a national infrastructure level construction, is particularly important in terms of security requirements. In this paper, based on the current potential dangers of docker containers in file isolation and the high security performance requirements in the construction, we design a docker container-based file system isolation scheme and prove through theoretical derivation that the scheme can effectively guarantee the isolation requirements of the file system in docker containers.

2. Docker System Architecture

Docker is a high-level container engine technology based on LXC (Linux Container), developed and opensourced by DotCloud, which is the main stream solution for virtualization processes [10]. LXC [11] is a kernel virtualization technology that provides lightweight virtualization for resource and process isolation. Docker and traditional virtualization technologies are both based on the isolation and reallocation of host resources, but docker’s virtualization solution is fundamentally different from the traditional ones.

2.1. Comparison of Docker and Traditional Virtualization Solutions. Among the traditional virtualization solutions (VMware, KVM), VMware is based on a fully virtualized design architecture, which is designed with the idea that the virtual machine virtualizes the entire underlying environment [12]. But KVM is an architecture based on hardware-assisted virtualization design, which implements virtualization operations through hardware-assisted support for processing sensitive instructions. Both of these make the operating system of the virtual machine completely independent from the host, and although this architecture virtual machines are very isolated, they are very heavy virtualization operations. In contrast, docker’s design architecture of host-level kernel-based virtualization allows for more lightweight virtualization. Figure 1 shows how the design architecture of docker container virtualization differs from the design architecture of traditional technologies [13].

2.2. Namespaces. The basis for the isolation between Linux containers, containers, and hosts is the namespace (NameSpace) in the Linux kernel. The mutual isolation of resources between different namespaces in Linux is the basis for container virtualization [14]. Linux supports the following six different namespaces [15].

1. UTS Namespace: system information such as host name, system version number, and system name is recorded in the UTS namespace, which enables the isolation of host names and domain names between containers, allowing each to have its own host name and domain name. This enables each container to be seen as an independent node in the overall network architecture.

2. Network Namespace: the role of this namespace is to achieve the isolation of the container’s network resources so that each container has its own independent network environment including IP routing tables, firewalls, and network devices.

3. PID Namespace: The role of this namespace is to achieve the isolation of process numbers between different containers, to ensure that the process numbers between containers are not visible to each other which is of great significance for resource isolation. Its tree structure is to achieve process number isolation, the same PID namespace processes that are visible to each other, cross-PID namespace process isolation from each other, and at the same time, to achieve the isolation of the processes in the container and the host process.

4. Mount Namespace: This namespace enables each container to have a separate file system and also enables the container to open a shared dire.

5. IPC Namespace: this namespace restricts access to the file system within containers and between containers and the host through the Mount namespace but still enables interaction between file systems by means of thread passages. IPC stands for thread communication, which is carried out in Linux systems by means of semaphores, shared memory, and message queues. The IPC namespace allows containers to have identifiers to isolate intercontainer process communication.

6. User Namespace: The namespace can achieve the isolation of security-related identifiers, mainly system user IDs, user group IDs related to user permissions, etc., to facilitate the management between their users and groups, and the root user in the container does not have special privileges on the
host, in order to achieve the security of the operation in the host.

3. Resource Isolation Design in Grid systems

(1) The construction of virtualization in the power grid system makes the virtualization of devices and clouding of applications become the current trend. How to design the file isolation scheme between applications deployed in different docker containers, and the file isolation scheme in different docker clusters is crucial to guarantee the secure operation of the file system in the power system. The file isolation mechanism of docker itself may have the problem of spanning interaccess to file resources between different containers in a complex grid system [16]. To address this problem, this paper proposes a file resource isolation scheme for docker containers in power grid systems.

3.1. Multilevel Container Resource Isolation Method Design. Before designing the scheme, we define the subject and object of the file isolation scheme. The subject of file isolation is different clusters \( R_i (1 < i < n) \), and docker containers \( C_j (1 < i < n) \) in different clusters. The object and isolation objects are the disk files in different clusters, and these disk files belonging to different subjects can be specifically divided into: disk files isolated between clusters \( SF_i (1 < i < n) \), disk files shared between clusters \( RF_i (1 < i < n) \), files inside containers \( CF_j (1 < i < n) \), and shared data volumes between containers and disks \( DF_j (1 < i < n) \).

In the text, cluster \((R, O)\) denotes the cluster environment, \( R = \{ R_1, R_2, R_3, \ldots R_n \} \), where \( R_i (1 < i < n) \) denotes the host node \( i \) in the cluster and \( R \) denotes the set of all hosts, where \( R_i = \{ D_1, D_2, D_3, D_4, \ldots D_n \} \). \( D_i (1 < i < n) \) denotes container \( i \), starting on the host. \( D \) is the set of containers on the host, and \( R \) is the set of isolated subjects; \( O = \{ R_1 F_{11}, R_1 F_{12}, \ldots R_1 F_{1j}, R_2 F_{21}, R_2 F_{22}, \ldots R_2 F_{2j}, \ldots R_n F_{11}, D_1 F_{1}, D_1 F_{2}, \ldots D_n F_{1}, D_n F_{2}, D_n F_{3}, D_n F_{4}, \ldots D_n F_{j} \} \), \( R_i F_j \) denotes the shareable file \( j (1 < i < n, 1 < j < n) \) on host \( i \), \( D_i F_j \) denotes the file \( j (1 < i < n, 1 < j < n) \) in container \( i \), and \( O \) denotes the set of isolated guests, and specifying the relation \( \text{READ} : R \rightarrow O \) denotes the set of readable files of \( R \), and \( \text{WRITE} : R \rightarrow O \) denotes the set of files that \( R \) can write; and using \( \text{AU} : R = P \) denotes the security level of \( R \), and \( \text{AU} : D = P \) denotes the security level of \( D \), where \( P \in \{ P_1, P_2, \ldots P_j \} \), and \( P_1 > P_2 \) denotes the \( P_1 \) security level above \( P_2 \).

3.2. Multilevel Container Resource Isolation Policy Design. The container file resource isolation policy is customized based on the subject-object relationship of container isolation above.

Policy 1. The range of reading and writable files for a container is the container’s own files and the shared files of the host to which the container belongs.

\[
\text{READ}(D) = \{ DF_j \cup RF_j \} \forall i < n, \quad \exists j 1 < j < n, \\
\text{WRITE}(D) = \{ DF_j \cup RF_j \} \forall i < n, \quad \exists j 1 < j < n.
\]

Policy 2. When multiple containers are at different security levels, when operating on their shared data volumes, only the highest security level container has write operations, and the other security level containers only have read operations.

\[
\text{AU}(D_i) > \text{AU}(D_j) \rightarrow \text{READ}(D_i) = \text{READ}(D_j) = \{ RF_i \} \ i 1 < i < n, \\
\text{AU}(D_i) > \text{AU}(D_j) \rightarrow \text{WRITE}(D_i) = \{ RF_i \} \exists i 1 < n, \text{WRITE}(D_i) = \{ 0 \}.
\]
other containers only have read permission, and the read operation does not change the file content, thus ensuring the security of the file content. Figure 2 shows the flowchart of isolated file identification.

4. Implementation of Resource Isolation in Grid Systems

In the cloud environment, various grid devices (such as energy meters, collection terminals, and digital meters) and related processes can rely on the docker deployment and use the above-designed multipole container resource isolation scheme to ensure that each software has an independent space, as well as the security of the files of containers with high security level during the interaction between different software. As shown in Figure 3, the containers of different hosts set the security level of the containers inside the host through the configuration file of the host, and the containers inside the host hook up their files to the host through the shared files with the host and set the security level between the internal files according to the security level of the containers. The host interacts its own files with the host configuration file with the highest security level in the cluster to realize the interaction of file systems in the cluster.

4.1. Container File Isolation Security Check. The operation of the interaction between the hosts in a multipole container system and between the hosts and containers is defined in a centralized manner. In grid systems, information read and write operations between devices always have a tendency to aggregate information from the bottom device to the top device, while the bottom device often does not have write access to the file system for the top device. The distribution of device instructions is often the higher-level devices to the bottom-level devices to communicate.

We set \( R_h \) as the host that represents the center of the cluster and implement the aggregation of other host files in the cluster and the files of containers in other hosts. The configuration in the global configuration file can set the security level of each host.

![Flowchart of isolated file identification.](image)

The security level of different containers inside the host is set by the host’s configuration file. Let there be containers \( D_1, D_2, D_3, D_4 \) inside the \( R_1 \) host. \( AU(D_1) > AU(D_2), AU(D_3) > AU(D_4), AU(D_2) > AU(D_3), \) and \( AU(D_2) > AU(D_1) \). According to strategy I and strategy II, it is obtained that

\[
\begin{align*}
\text{READ}(D_4) &= \{R_1F_4, R_1F_1\}, \\
\text{WRITE}(D_4) &= \{R_1F_4\}, \\
\text{READ}(D_3) &= \{R_1F_3, R_1F_2, R_1F_1\}, \\
\text{WRITE}(D_3) &= \{R_1F_3, R_1F_2\}, \\
\text{READ}(D_2) &= \{R_1F_2, R_1F_3, R_1F_1\}, \\
\text{WRITE}(D_2) &= \{R_1F_2, R_1F_3, R_1F_4\}, \\
\text{READ}(D_1) &= \{R_1F_1, R_1F_2, R_1F_3, R_1F_4\}, \\
\text{WRITE}(D_1) &= \{R_1F_3, R_1F_4\}.
\end{align*}
\]

From the above derivation, we can see that the container with the highest security level can modify its own shared files mounted on the host and the shared files on the host of other containers with lower security level than itself. Containers with lower security level cannot modify the shared files, thus ensuring the isolated security of the shared files of the container with the highest security level. The other security level containers can read the shared files of the containers and write the files of the lower security level containers to ensure that the higher security level containers can distribute commands to the lower security level containers.

4.2. Host File Isolation Security Check. Let the security level between different hosts be
From the above documented relationship, it follows that

\[
\begin{align*}
& \text{READ}(R_0) \cap \text{READ}(R_1) = [R_1F_1, R_1F_2], \\
& \text{READ}(R_0) \cap \text{READ}(R_2) = [R_2F_1, R_2F_2], \\
& \text{READ}(R_0) \cap \text{READ}(R_3) = [R_3F_1, R_3F_2], \\
& \text{READ}(R_0) \cap \text{READ}(R_4) = [R_4F_1, R_4F_2], \\
& \text{WRITE}(R_0) \cap \text{WRITE}(R_1) = [R_1F_1], \\
& \text{WRITE}(R_0) \cap \text{WRITE}(R_2) = [R_2F_1], \\
& \text{WRITE}(R_0) \cap \text{WRITE}(R_3) = [R_3F_1], \\
& \text{WRITE}(R_0) \cap \text{WRITE}(R_4) = [R_4F_1].
\end{align*}
\]

Then from the above, file relationship can be obtained that the low security level of the host can read the corresponding file from $R_0$, to achieve the function of instructions, while part of the high security level of the file low security level of the host does not have permission to access, in the write operation level $R_0$ can be read and write to all the low level of the host in the file, so as to achieve the cluster file information will flow and the function of instructions. At the same time, the data gathered can be saved in a file that is accessible only by itself.
The access to each file system in the cluster is ensured by a two-tier file isolation policy with a file isolation policy between containers within the hosts and a file isolation policy between hosts.

5. Conclusion

In recent years, the spreading dynamics of computer viruses as a popular research in network science has entered a period of rapid development. Through dynamic system modeling, we can explore the propagation law of computer viruses and provide a formulating countermeasures.

By deploying the applications in the grid system with docker containers, using the file scheme designed in this paper based on the isolation feature of docker, we can effectively ensure the independence and security of the file system between different containers, realize the security level of containers and hosts according to the security level set by the configuration file, and realize the file isolation of containers and hosts with different security levels. By achieving isolation between containers of different security levels, the spread of viruses between containers is thus isolated to a certain extent and the grid system is protected. However, when the grid system is not deployed as docker, the isolation scheme proposed in this paper will not be possible.

At the same time, in further exploration, more than just a file system isolation will be carried out; the design of an isolation method for system operation rights has not been explored in the current study, and in future exploration, the isolation of operation rights, combined with the isolation of the file system studied in this paper, can be achieved to further protect the security of the power system.

Data Availability

The datasets used in this study are available from the corresponding author upon request (zhengfeng@bupt.edu.cn (Feng Zheng)).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was funded by the State Grid Corporation Headquarters Management Technology Project, Project Name: Research on Key Technologies of Lightweight Embedded Operating System for Intelligent Measuring Equipment, Project Code: 5700-202055484A-0-0-00.

References

[1] C. Liu, J. Wu, C. Long, and D. Kundur, “Reactance perturbation for detecting and identifying FDI attacks in power system state estimation,” IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 4, pp. 763–776, Aug. 2018.
[2] Z. Qu, Y. Zhang, N. Qu, L. Wang, Y. Li, and Y. Dong, “Method for quantitative estimation of the risk propagation threshold in electric power CPS based on seepage probability,” IEEE Access, vol. 6, pp. 68813–68823, 2018.
[3] Y. Li, Z. Li, and L. Chen, “Dynamic state estimation of generators under cyber attacks,” IEEE Access, vol. 7, pp. 125253–125267, 2019.
[4] W. Kang, P. Zhu, and X. Liu, “Integrated Defense Mechanism based on attack goals against three Attack Strategies in smart grid,” in Proceedings of the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 1027–1032, Toronto, ON, Canada, July 2020.
[5] T. R. Sharafeev, O. V. Ju, and A. L. Kulikov, “Cyber-Security Problems in Smart Grid Cyber Attacks Detecting Methods and Modelling Attack Scenarios on Electric Power Systems,” in Proceedings of the International Conference on Industrial Engineering, Applications and Manufacturing (ICEIAM), pp. 1–6, Moscow, Russia, May 2018.
[6] G. Z. Y. G. Yufeng Zhao and L. Cheng, “Design and development of power equipment simulation cloud platform based on container technology,” Computer Engineering, vol. 47, pp. 171–177, 2021.
[7] R. R. M. A. Krasnov and D. M. Wilborne, Data Storage Security in Docker, SoutheastCon, Atlanta, Georgia, 2020.
[8] A. Ahmed and G. Pierre, "Docker Image Sharing in Distributed Fog infrastructures,” in Proceedings of the 2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), Sydney, NSW, Australia, December 2019.
[9] G. W. Bei Wang, "Analysis and research on realizing electricity sales platform based on microservice architecture,” Electric Power Information and Communication Technology, vol. 17, 2019.
[10] J. G. Yong Ma and P. Wang, “Research on application resource pool technology based on docker,” Electric Power Information and Communication Technology, vol. 16, 2018.
[11] A. Kovcs, “Comparison of Different Linux Containers,” in Proceedings of the 2017 40th International Conference on Telecommunications and Signal Processing (TSP), Barcelona, Spain, July 2017.
[12] S. Varrette, M. Guzek, and P. Bouvry, “Hpc Performance and Energy-Efficiency of Xen, Kvm and Vmware Hypervisors,” in Proceedings of the 2013 25th in- ternational Symposium on Computer Architecture and High Performance Computing, Porto de Galinhas, Brazil, October 2013.
[13] X. Gao, M. Kayaalp, Z. Gu, and H. Wang, "Containerleaks: Emerging Security Threats of Information Leakages in Container clouds,” in Proceedings of the 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Denver, CO, USA, June 2017.
[14] M. Bian and L. Wang, "Analysis and Research on the Vulnerability of Docker Container in cloud environment,” Information Security, vol. 20, pp. 85–95, 2020.
[15] C. X. Bin Lv, "Docker container security analysis and enhancement program research,” Information Network Security, vol. 1, pp. 15–22, 2021.
[16] J. F. Lanfang Ren and X. Zhuang, "Research on docker container security protection technology,” Telecommunications Engineering Technology and Standardization, vol. 33, 2020.