Research on Security Issues of Docker and Container Monitoring System in Edge Computing System

Xinchen Xu1, *, Aidong Xu2, Yixin Jiang2, Zhen Wang1, Qianru Wang1, Yunan Zhang2 and Hong Wen1

1 School of Aeronautics and Astronautics, University of Electronic Science and Technology of China (UESTC), Chengdu 611731, China
2 Electric Power Research Institute, China Southern Power Grid Co., Ltd. Guangzhou, China
Email: uesityxxc@outlook.com; sunlike@uestc.edu.cn
*Corresponding Author Email: 201822100212@std.uestc.edu.cn

Abstract. With the development of the 5G technology, edge computing will sink the computing center to the edge of the network close to the user, with low latency, high security, and lightweight features. The development of edge computing also puts forward a series of requirements for security, standardization, and unification. Docker container technology highly meets the existing needs of edge computing due to its lightweight, standardization, convenience, and security isolation. This article mainly analyzes the vulnerability of Docker containers and summarizes the security problems faced by the application of Docker container technology under edge computing systems. It also introduces a container monitoring software Prometheus and proposes a feasible edge computing risk monitoring model based on the Docker engine and Prometheus monitoring software.

1. Introduction
Edge computing (EC) was formally proposed in 2016 in [1], in which the definition of edge computing was given. Edge computing refers to the technology of computing at the edge of the network. The edge is defined as any computing and network resource node between the data source and the data center [1]. With the development of 5G technology, edge computing has received widespread attention due to its low latency, security, and lightweight characteristics. Edge computing technology has been applied in various fields, such as smart cities, smart grids, the Internet of vehicles, the Internet of Things (IoT) EC also provides a performance platform for some new technologies, such as physical layer security technologies [12]. Literature [2] proposed an edge computing system based on smart grid to overcome the existing problems in traditional grid systems. Insufficient, it fully realizes the requirements of high bandwidth and low latency. Document [3] proposed an optimized radio frequency fingerprint classification algorithm based on the low latency and high security characteristics of edge computing under the Internet of Things system. Literature [5] introduced a meter authentication scheme in the smart grid. According to the characteristics of edge computing close to the terminal, a physical layer auxiliary information authentication scheme was designed to reduce the time delay of terminals access authentication.

However, the development of edge computing also faces many challenges[8]. First of all, edge computing devices need to process and coordinate a large number of computing resources upward, and there are massive heterogeneous terminals and massive data access downward. Because it is close to the terminals, it is more vulnerable to terminals attacking. Edge computing devices need to provide
more secure protection measures for applications running on it. Secondly, although the Edge Computing Equipment Industry Consortium (ECC) has successively released “Edge Computing Reference Architecture 2.0” and “Edge Computing Reference Architecture 3.0”[15], the development of edge computing in various fields still lacks a unified specification. Different edge computing devices, heterogeneous data storage methods and diversification of operating systems have brought great resistance to software application development. A large number of heterogeneous data storage, application upgrades, and software migration between different edge devices are all facing difficulties.

Docker is a lightweight container engine based on LXC, a virtualization technology that uniformly packages the underlying files, images, and applications [4]. It has the advantages of lightweight, convenience, standardization, and security isolation, which highly meets the existing needs of edge computing [10]. Different programs can be run on any host machine with a Docker engine only after a unified Docker package. Multiple independent system container environments can be provided on a single operating system by encapsulating the environment and dependent packages required by the software run time. Docker technologies are built on the LXC technology of the Linux kernel. Due to its opening system, the LXC technology let Docker system be easy to attack. This article mainly analyzes the security problems faced by the application of Docker container under edge computing systems and introduces a container monitoring mechanism to perform edge computing risk monitoring model based on the Docker engine and Prometheus monitoring software.

2. Docker Security Analysis

Docker technology can be divided into three parts--Docker image, Docker container and Docker registries. The Docker registries delivers the Docker image to the local host according to the requirements. The local host builds Docker containers for different systems based on the “sandbox” mechanism. The Docker kernel reuse method provides a unified operating specification for applications developed in multiple operating systems and different languages [6]. Docker uses Linux underlying functions such as (Namespace component and Cgroups component) to complete the isolation between containers, processes, and systems, which provides extra security protection for the system. Typical examples are the CORD project of the Open Network Foundation and the AkrainoEdge Stack project of the Linux Foundation. However, the security of Docker technology itself depends on the LXC technology of the Linux kernel, which is an open resource system and makes Docker system vulnerable. The security issues of Docker in the edge computing system can be divided into three parts: Docker container security, Docker image security and host security.

2.1. Docker Container Security

The Docker container security is combined by two parts: Internet security and resource security. Firstly, the network in the Docker container communicates with the host through a virtual bridge, but the virtual bridge in Linux system only has a forwarding function for incoming and outgoing traffic without filtering. Attackers may send a large number of illegal data packets to cause MAC Flooding attacks. A feasible solution to this problem is to filter network traffic [7]. Secondly, the principle of Docker for process isolation is to use the Namespace component under Linux. By using different naming to complete the isolation between host names and domain names between different containers, each container can be regarded as a separate node by other hosts. When users are not familiar with the use of Docker or have not enabled the Cgroups component for resource restriction, attackers can create a large number of useless containers to occupy computing resources and form a DoS attack [9]. The common solution to this problem is to reduce the permissions of Docker users, prohibit mounting, and prohibit access to sensitive files, but this also reduces the scalability of the container. At the same time, the Cgroups component does not have a disk resource restriction function. In order to fully protect against Dos attacks, we can also monitor the usage of system resources through the container monitoring system to determine the risk of the container.

2.2. Docker Image Security

Docker image sources are mainly public image sources such as Docker hub and aliyun. Some users may also download images through private image repository. Regardless of the commonly used public
mirror warehouses or private mirror warehouses, other users are basically allowed to upload their own created mirrors. When the downloaded image is defective or has a backdoor after malicious tampering, the image has certain security risks [13]. Users can conduct security protection by establishing private mirror warehouses and checking Docker mirror patches regularly.

2.3. Host Security
The security of the host is mainly reflected in the security of the kernel. Docker essentially isolates the file system between the container and the host through the Mount Namespace under Linux. The file system structures in different Namespaces do not affect each other, which results in the root user in the container having the same permissions as the root user in the host. The user privilege escalation operation in the container may cause the user privileges in the host to change, thereby harming the entire host and all containers. The sandbox mechanism does not isolate all system files. Containers share key system information in some root directories of the Linux system, and permission changes may cause information leakage[11]. In order to deal with this problem, a feasible solution is to establish the minimum permissions that allow the container to run and prohibit the container user performing root operations.

3. Functions of Prometheus
Commonly container protection measures include restricting permissions, prohibiting mounting, prohibiting access to sensitive system files, and traffic filtering. However, on the one hand, this series of measures will reduce the scalability of the container, on the other hand, these methods rely on the components that come with the Linux system and have some drawbacks. The container monitoring system can be used as a basic safety measure to provide safety protection for containers, make up for the defects of commonly used container protection measures and indirectly detect the safety risks of container operation through various indicators [14]. Common container monitoring software include Docker stats, cAdvisor, Sysdig, Promethues and other open source software. Here is a brief introduction to container monitoring system Prometheus.

Prometheus is an open source project developed in Go language. Its working principle is to periodically capture the status of the monitored component through the Http protocol, and any component can access monitoring as long as it provides an http interface. Therefore, Prometheus has high scalability, and it also has many other advantages: support flexible query language, single node independent work, store data with time series database and support chart visualization interface configuration. The specific working process of Prometheus is shown in Figure 1.

![Figure 1. Working process of Prometheus.](image)

Prometheus includes three core components, which are server, alertmanager and pushgateway. The server component is mainly responsible for data collection and storage, and provides PromQL language for query. The alertmanager component provides alarm management. When a certain
monitoring indicator in the container exceeds a set threshold, an alarm email can be sent to the target mailbox by email. Pushgateway is used to collect the data cache of temporary nodes. When some nodes exist for a short period of time, the Prometheus server cannot capture the data. Pushgateway can push the indicators to the gateway for cache and upload them together when other indicators are collected.

4. Edge Computing Risk Monitoring Model

Based on the Docker engine and the Prometheus monitoring system, an edge computing risk monitoring model is built from the edge side to the user side. The model figure is shown in Figure 2. In this model, both the edge side and the user side use the Docker engine as a support system, and a private Docker image warehouse is established externally to store and send the Docker image to the edge computing device and the user side. The software applications on the user side are run using Docker containers to isolate them from the system. The cAdvisor component collects various indicators running inside the container to determine the running status of the Docker container. Prometheus is installed on the edge computing device to collect and store the container metrics of each client. When the metric crosses the threshold, alertmanager sends an alert email to the designated mailbox. The control unit on the edge computing device uses the content of the alert email to rate the risk of the Docker container, and issues instructions to the client to request the restart, shutdown, or deletion of the Docker container at high security risk.

![Diagram of Edge Computing Monitoring Model](image)

**Figure 2.** Edge computing monitoring model.

The monitoring indicators established by the Prometheus software are shown in Table 1.

| Index          | Threshold value | Duration |
|----------------|-----------------|----------|
| Disk usage     | 80%             | 1 min    |
| Memory usage   | 70%             | 1 min    |
| Break Down     | 0               | 5 s      |
| Network_bandwidth | 100Mb/s   | 3 min    |
| CPU usage      | 60%             | 1 min    |
| I/O usage      | 60%             | 1 min    |
| TCP_established| 1000            | 5 s      |
Prometheus also supports to connect with the open source software Grafana which can display various indexes and charts. The interface of Prometheus monitoring model in edge side is shown as Figure 3.

![Prometheus Monitoring Interface](image_url)

**Figure 3.** The interface of Prometheus monitoring model.

5. Conclusions
This article demonstrates the advantages and disadvantages of using Docker containers under the edge computing system. Different programs are run on the different Docker progresses which provides extra security protection for the system. However, the LXC technology of the Linux kernel that Docker technologies depend on is an opening system, which makes Docker system be easy to attack. By taking advantages of the container monitoring software Prometheus, a container risk monitoring model from the edge side to the terminal sides is built, which can monitor abnormal behavior of the system and send alarm emails, and even shutdown, or delete the Docker container at high security risk. The future research direction is to add the cloud side to the model and the edge side can be regarded as the monitoring object of the cloud side to establish a more complete three-tier monitoring system.

6. Acknowledgments
This work is supported by the National major R & D program (No. 2018YFB0904900, 2018YFB0904905).

7. References
[1] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637-646, Oct. 2016.
[2] S. Chen, H. Wen, Wu J, et al., "Internet of Things based Smart Grids Supported by Intelligent Edge Computing," *IEEE Access*, vol. 7, pp. 74089-74102, 2019.
[3] F. Xie et al., "Optimized Coherent Integration-Based Radio Frequency Fingerprinting in Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3967-3977, Oct. 2018.
[4] J. Bhimani et al., "Docker Container Scheduler for I/O Intensive Applications Running on NVMe SSDs," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 313-326, 1 July-Sept. 2018.
[5] H. Wen, Y. Wang, X. Zhu, J. Li and L. Zhou, "Physical layer assist authentication technique for smart meter system," *IET Communications*, vol. 7, no. 3, pp. 189-197, 12 February 2013.
[6] S. Kwon and J. Lee, "DIVDS: Docker Image Vulnerability Diagnostic System," *IEEE Access*, vol. 8, pp. 42666-42673, 2020.
[7] H. Wen, P. Ho and B. Wu, "Achieving Secure Communications over Wiretap Channels via Security Codes from Resilient Functions," *IEEE Wireless Communications Letters*, vol. 3, no. 3, pp. 273-276, June 2014.
[8] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81-84, Sept. 2014.

[9] J. Xiao, H. Wen, B. Wu, X. Jiang, P. Ho and L. Zhang, "Joint Design on DCN Placement and Survivable Cloud Service Provision over All-Optical Mesh Networks," *IEEE Transactions on Communications*, vol. 62, no. 1, pp. 235-245, January 2014.

[10] L. Hu et al., "Cooperative Jamming for Physical Layer Security Enhancement in Internet of Things," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 219-228, Feb. 2018.

[11] B. Ji, Y. Li, B. Zhou, C. Li, K. Song and H. Wen, "Performance Analysis of UAV Relay Assisted IoT Communication Network Enhanced With Energy Harvesting," *IEEE Access*, vol. 7, pp. 38738-38747, 2019.

[12] Y. Xie, H. Wen, B. Wu, Y. Jiang and J. Meng, "A Modified Hierarchical Attribute-Based Encryption Access Control Method for Mobile Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 383-391, 1 April-June 2019.

[13] L. Hu, H. Wen, B. Wu, J. Tang and F. Pan, "Adaptive Secure Transmission for Physical Layer Security in Cooperative Wireless Networks," *IEEE Communications Letters*, vol. 21, no. 3, pp. 524-527, March 2017.

[14] S. Fu, B. Wu, H. Wen, P. Ho and G. Feng, "Transmission Scheduling and Game Theoretical Power Allocation for Interference Coordination in CoMP," *IEEE Transactions on Wireless Communications*, vol. 13, no. 1, pp. 112-123, January 2014.

[15] Y. Mao, J. Zhang and K. B. Letaief, "Joint Task Offloading Scheduling and Transmit Power Allocation for Mobile-Edge Computing Systems," *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, CA, 2017, pp. 1-6.