Vulnerability Analysis and Security Research of Docker Container

Jiang Wenhao[1], Li Zheng[2]
[1]Department of Computer Engineering, Chongqing Aerospace Polytechnic, Chongqing, China
[2]Mobile Communication Engineering Research Center, Chongqing University of Posts and Telecommunications, Chongqing, China
yljdp@yeah.net

***Abstract*—Docker is an open source application container engine that enables users to pack applications and rely on packages to portable containers. However, Docker also has concerns about security. This paper starts from four aspects of Docker vulnerability, including file system isolation, process and communication isolation, device management and host resource constraints, network isolation and image transmission. Interact with the security module of Linux kernel to enhance the security of Docker, and take active and effective measures to enhance the security of Docker. This paper presents a general picture of current Docker security research, explores and looks into the development trend of Docker security research, and lays a good foundation for the better application of Docker in production.**

Keywords—Docker, vulnerability, kernel security

I. DOCKER INTRODUCTION

Cloud computing, which can provide flexible and flexible services, has the characteristics of rapid deployment and portability, and occupies most of the market share in today's computing field[1]. Among them, resource virtualization technology [2] plays a vital role in realizing the on-demand allocation of cloud computing infrastructure services. The traditional hardware level virtualization such as Vmware, Xen, KVM and Microsoft Hyper -v has to reduce a resource utilization ratio by virtualying a complete operating system layer. In order to find a more efficient isolation technology alternative, Docker has launched some open source container project Docker[3] in March 2013. In order to find more efficient isolation technology alternatives, Docker has been successful and widely used in various scenarios. It's wide application makes the research on Docker security more urgent and practical.

Docker[3] is an open source engine. Docker that automatically deploys the development application to the container. In the container execution environment of virtualization, an application deployment engine is added. The goal of the engine is to provide a lightweight and fast environment, which can run the developer's program and deploy the program from the developer's notebook to the test environment conveniently and efficiently. And then deploy to the production environment.

A. Vulnerability analysis of Docker host

Docker uses client server (client-server) schema. Docker client communicates with Docker daemon, Docker daemon handles complex and heavy tasks, such as building, running, publishing Docker container. Docker client and daemon can run on the same system. You can also use the Docker client to connect a remote Docker daemon. Docker client and daemon to communicate via socket or RESTful API[4].Docker Client is used as a communication client to communicate with Docker Daemon daemon, so as to manage Docker.

The Docker schema is shown in Figure 1:

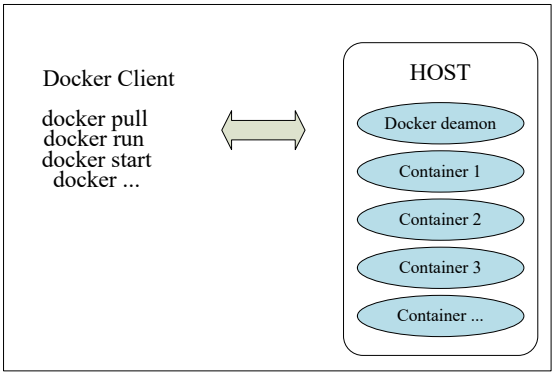

Fig. 1. Docker schema diagram

B. Docker Remote API and DockerSwarm

Docker Remote API[5] is the REST API that takes the place of rcli (remote command line interface), which helps to send requests, obtain and send data, and retrieve information.

Swarm is a relatively simple tool launched by Docker in early December 2014 to manage Docker clusters. Swarm transforms a group of Docker host[6] into a single, virtual host. Swarm using the standard Docker API interface as its front-end access entry, that is, various forms of Docker Client (Docker, Docker),Docker and so on can directly communicate with Swarm, and the structure diagram of Swarm is shown in Figure 2.

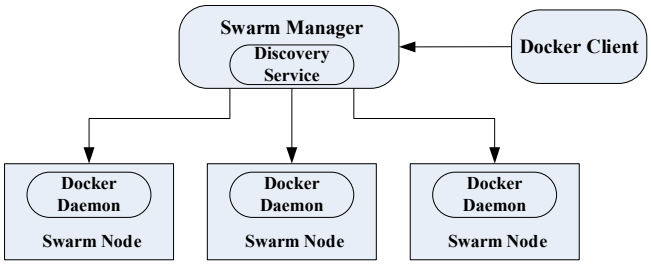

Fig. 2. structure diagram of Swarm

II. Docker Vulnerability Analysis

Docker vulnerability is mainly reflected in four aspects: file system isolation, process and communication isolation, device management and host resource limitation, network isolation and image transmission. Research on vulnerability of Docker can provide a basis for finding targeted security enhancement measures.

A. File system isolation

The realization of isolation file function in Docker is based on Mount namespace. Different files[7] are stored in different namespaces to avoid the interaction among file structures. However, the root permissions in the container are similar to those of host root users, especially when RW mount is installed in the host computer and the container has root privileges. Once the container executes "Chmod a +s[program file]", the users who run the program can get root permissions, thus greatly increasing the probability of leakage of related information.

B. Remote isolation and communication isolation

Docker uses PID namespace[8] to isolate the process, separate the PID. of the process, and separate the PID namespace of the process. There is a separate counting program. The system kernel realizes the maintenance of PID namespace through the tree, where the root namespace is at the top level, and the parent node can get the child node information, but the child node can't get the parent node information. However, it needs to be clear: when it is 1The PID is terminated, which results in a complete stop of the container and denial of service.

C. Device management and host resource constraints

The isolation of domain name and hostname in Docker container is realized by UTS namespace, while Cgroups[9] is used to restrict the use of resources. However, by default, it is closed, and it is easy to increase the probability of exhaustion of equipment resources, network bandwidth and memory space. In addition, taking into account that users can get resources in the container, user defined code is not properly restricted. It will cause host to refuse service, especially when using SaaS, PaaS and IaaS, should pay enough attention to prevent the occurrence of denial of service.

D. Network isolation and image transmission

Network namespaces in Docker are responsible for isolating network resources, such as IP routing table, TCP/IP protocol, isolation of network devices, etc. However, by default, Docker specifies IP and Network namespaces for containers, and then connects with bridges. This way is only forwarding traffic, without filtering, will face greater MAC Flooding risk. When mirroring is transmitted, Docker does not check the mirroring of the pull back. If some data are replaced, there may be a man in the middle attack.

III. Docker and Kernel Security System

The existence of some kernel security systems can be used to enhance the security of Docker and kernel, including the Linux function[10] and the Linux security module (LSM). The Linux function restricts the permissions assigned to each process. LSM provides a framework that allows the Linux kernel to support different security models. LSM, which has been integrated into the official Linux kernel, includes AppArmor, SELinux and AppArmor.

A. Linux function

The Docker container runs on the kernel shared with the host system, so most tasks can be handled by the host. Therefore, in most cases, it is not necessary to provide full root permissions to the container. Therefore, deleting some root functions from the container will not affect the availability or function of the container, but also effectively enhance the security of the system. For example, CAP_ can be removed from the container to modify the network function of the system. NET_ADMIN function, because all network configurations can be handled by the Docker daemon before launching the container.

Docker allows configuring functions that can be used by the container. By default, even if the intruder obtains root access rights in the container, Docker can also prohibit a large number of Linux functions from its container to prevent the intruder from damaging the host system. Some of the functions disabled by the. Docker container are shown in TABLE I, which can be found in the Linux manual page.

TABLE I Partial Functions Disabled By Docker Container

Docker process name	Disabled functions
CAP_SETPCAP	Modifying process functions
CAP_SY S_MODULE	Insert / delete kernel module
CAP_SY S_RAWIO	Modify kernel memory
CAP_SY S_PACCT	Configuration process record
CAP_SY S_RESOURCE	Covering resource constraints
CAP_SY S_NICE	Modify priority of process
CAP_SY S_TIME	Modify system clock
CAP_AUDIT_WRITE	Writing audit logs
CAP_MAC_OV ERRIDE	Ignore kernel MAC policy
CAP_SY SLOG	Modifying kernel printk behavior
CAP_SY S_TTY_CONFIG	Configuring TTY devices
CAP_AUDIT_CONTROL	Configuring audit subsystem
CAP_MAC_ADMIN	Configure MAC configuration
CAP_SY S_ADMIN	Select all
CAP_NET_ADMIN	configure network

B. AppArmor

AppArmor[11] is also a Linux security enhancement model based on mandatory access control (such as SELinux), but it limits the scope to a single program. It allows administrators to load security configuration files into each program, which limits the function of the program.

On AppArmor enabled systems, Docker provides an interface to load predefined AppArmor configuration files when launching new containers. The configuration file will be loaded into the container in a mandatory mode to ensure that the process in the container is restricted by the configuration file. If the administrator does not specify the configuration file when starting the container, Docker daemon will automatically load the default configuration file into the container, which will refuse access to important file systems on the host, such as /sys/fs/cgroups/ and /sys/kerne/security/.

C. Capability mechanism

The ability mechanism makes the Linux permission more finely divided, changing the phenomenon of all privileges of root users in the past. The ability mechanism subdivides the privilege into 37 different abilities, covering

the operation rights of documents, processes, networks, systems, devices and so on. Docker provides limited capacity for containers under the condition of silent recognition. Users can also add or reduce capability privileges at runtime, depending on actual needs. The problem with ability mechanism is that the division of power is still not enough in some respects, and it can't meet the requirement of container restriction very well, CAP_SYS_ADMIN covers a lot of privileges and needs to be further subdivided. It is difficult to know which privileges are needed in containers when assigning permissions for containers. For this reason, the default capability permission list provided by Docker contains a broader scope. From a security point of view, it is not a good strategy.

D. Seccomp

Seccomp[12] can restrict system calls of user processes and filter system call parameters. System calls are the most important interface between user state and kernel state. By limiting it, the process can be guaranteed to operate in a safe and controllable range to a large extent. Docker default is based on whitelist mechanism, which uses Seccomp configuration file to specify which system calls are allowed, and more than 50 systems will be blocked. Some system calls are also blocked under specific parameters. Most of the containers can run normally under the configuration file.

E. mandatory access control

Mandatory access control can provide more stringent protection than discretionary access control (DAC) [13]. Because users can't change their security level and object's security attributes, they can only make objective and specific decisions by MAC according to the predefined user and data security level matching results. In this way, the user's subjective factors can be shielded and the security of the system can be better protected. In the Linux system, mandatory access control is usually used in conjunction with DAC, and the specific security modules are implemented under the framework of LSM (Linux security module). The security modules that have been implemented and merged into the kernel mainline are SELinux, AppArmor, Yama, SMACK and Tomoyo. These security modules can provide enhanced security protection for Docker containers. In addition to Yama, there will be mutual exclusion among modules. In general, different security modules will be selected according to the different distribution of Linux. Debain series usually use their own AppArmor, and Redhat series releases the SELinux security module.

The main problem of SELinux is that the operation is too complex. Developers and actual production departments are boycotting the use of SELinux and are incompatible with BTRFS drivers. The uniqueness of AppArmor is that it does not pay attention to the security of the whole system, but only provides mandatory access control for the specified process. Other processes are working in an uncontrolled state. The LiCShield security framework is used to consolidate Linux containers. The framework automatically generates a AppArmor configuration file by tracking and analyzing the required permissions of the container to ensure the safe and controllable operation of the container. Besides these security modules, users can also design more suitable security modules according to the LSM framework.

F. Network framework

There are some network frameworks in the Linux kernel that can be used to solve the Docker network security problems. This includes the Netfilter[14] framework and TC (tracfic controller) framework. Netfilter, which focuses on firewall data filtering, while TC focuses on traffic bandwidth control. Docker can control the communication between containers by modifying the configuration file to enable Iptable firewall and configuring firewall strategy. The TC can limit the container bandwidth by restricting the flow of virtual network interface on the host computer through the TC, or tagging the network packets with the Cgroups subsystem, and using the classified queue in the TC to restrict the traffic flow of the container, thus effectively preventing the exclusive network bandwidth type denial of service attack.

G. Integrity protection

The integrity protection technology in the Linux kernel can be well applied to Docker to ensure the integrity of mirroring and container running. The technologies involved include trusted computing open standard (Trusted Computing Group, TCG) the dm-verity sub module that implements integrity check function in some modules of Linux and Device Mapper. These technologies can be used to achieve integrity protection of Docker images.

H. Log audit

Docker although there is a relatively systematic implementation plan for visual audit of logs, there is still a need for further research on large-scale log data mining and automatic analysis. In terms of operational environment audit, a security assessment document developed by Docker company and CIS (Center for Internet security) can be referred to, and its specific assessment implementation tool is Docker bench for security.

I. Security threat detection

In the aspect of security threat detection for Docker, although traditional host-based threat detection and data flow-based threat detection technology can be adopted, it is still necessary to further enhance the application of Docker in combination with the characteristics of the Docker. This method belongs to an example of applying traditional host threat detection technology to Docker specific environment.

J. Streamline operation system

From a different perspective, there are many researches on the security issues of Docker. For example, in order to provide a safer container running environment, a batch of streamlined operating system projects, including CoreOS, Ubuntu Core, have been put forward, which are more suitable for container operation. A new lightweight architecture scheme is proposed, which will enable the application to run directly on the bare machine with the help of customized kernel, and no longer share the kernel system with each other. The representative solutions include Unikernels and so on. Catuogno compares and compares a security model proposed by CCRA (common criteria recognition arrangement), such as container safety standard model and Reshetova, under Docker environment, and points out the equivalence of the two security models in essence. Sharath N and so on introduced how to use the

Stackelberg model to find a best combination point which can take into account Docker security and ensure operation efficiency through linear programming method.

K. Other related safety enhancement measures

Docker although there are more systematic implementation plans for visual audit of logs, there is still a need for further research on large-scale log data mining and automatic analysis. In terms of operational environment audit, a security assessment document developed by Docker company and CIS (Center for Internet Security) can be referred to, and its specific assessment implementation tool is Docker bench for security.

In the aspect of security threat detection for Docker, although traditional host-based threat detection and data flow-based threat detection technology can be adopted, how to combine Docker's own characteristics for targeted applications needs to be further strengthened. Abed A S and so on have introduced in detail how to pass the system call to detect the Docker process, and cluster analysis of the defined call frequency. This method belongs to an example of applying traditional host threat detection technology to Docker specific environment.

From a different perspective, there are many researches on the security issues of Docker. For example, in order to provide a safer container running environment, a batch of streamlined operating system projects, including CoreOS, Ubuntu Core, have been developed, which are more suitable for container operation. A new lightweight architecture scheme is proposed, which will enable the application to run directly on the bare machine with the help of customized kernel, and no longer share the kernel system with each other. The representative solutions include Unikernels and so on. Catuogno compares and compares a security model proposed by CCRA (Common Criteria Recognition Arrangement) [15], such as container safety standard model and Reshetova, under Docker environment, and points out the equivalence of the two security models in essence. Sharath N et al. Introduced how to use Stackelberg model (Stackelberg Model) through linear programming to find a best combination of Docker security and operation efficiency.

IV. CONCLUSION

With the increasing popularity of container applications, more and more sensitive applications will be migrated to containers. How to enhance the security of containers will become a growing concern while providing convenient deployment and high resource utilization. First of all, this paper briefly introduces the main structure of Docker, and then combs the vulnerability of Docker. It points out that it is mainly embodied in four aspects: file system isolation, process and communication isolation, device management and host resource limitation, network isolation and image transmission. Then, the Docker and kernel security system are summarized, and the current Docker security research measures are summarized. Compared with traditional virtual technology, the security of the Docker container still needs to be improved. The two containers are not relative. They also play a complementary role in the same virtual isolation effect. Trying to make up the short board of Docker security will give full play to the advantages of Docker in virtual containers.

ACKNOWLEDGEMENT

The Ministry of Education Science and Technology Development Center, University Industry Research Innovation Fund, is a new generation of information technology innovation project: Vulnerability Analysis based on Docker Container and Reinforcement of Security Audit (project number: 2018A01008)

REFERENCES

[1] Bui T . Analysis of Docker Security[J]. Computer ence, 2015.

[2] Combe T , Martin A , Pietro R D . To Docker or Not to Docker: A Security Perspective[J]. Cloud Computing IEEE, 2016, 3(5):54-62.

[3] A R Manu, Jitendra Kumar Patel, Shakil Akhtar. A study, analysis and deep dive on cloud PAAS security in terms of Docker container security[C]// International Conference on Circuit. IEEE, 2016.

[4] Holger Gantikow, Christoph Reich, Martin Knahl. Providing Security in Container-Based HPC Runtime Environments[C]// International Conference on High Performance Computing. Springer, Cham, 2016.

[5] Yasrab R . Mitigating Docker Security Issues[J]. 2018.

[6] Garg S , Garg S . Automated Cloud Infrastructure, Continuous Integration and Continuous Delivery using Docker with Robust Container Security[C]// Automated Cloud Infrastructure, Continuous Integration & Continuous Delivery Using Docker with Robust Container Security. 2019.

[7] Sever D , Kisasondi T . Efficiency and security of docker based honeypot systems[C]// International Convention on Information & Communication Technology, Electronics & Microelectronics. 2018.

[8] Fuling L I , Haoxian H E . Implementation of online evaluation system security based on Docker[J]. Journal of North China Institute of Science and Technology, 2018.

[9] Chun-Lin L I , Zheng-Jun L , Ye W , et al. ModifiedConstruction Method of Docker-based Network Security Experimental Platform[J]. Communications Technology, 2019.

[10] Fotis LoukidisAndreou, Ioannis Giannakopoulos, Katerina Doka, etc. Docker-Sec: A Fully Automated Container Security Enhancement Mechanism[C]// IEEE International Conference on Distributed Computing Systems. IEEE Computer Society, 2018.

[11] Zerouali A , Mens T , Robles G , et al. On The Relation Between Outdated Docker Containers, Severity Vulnerabilities and Bugs[J]. 2018.

[12] Chin - Wei Tien, Tse - Yung Huang, Chia - Wei Tien, et al. KubAnomaly: Anomaly detection for the Docker orchestration platform with neural network approaches[J]. Engineering Reports, 2019, 1(5).

[13] Martin A , Raponi S , Combe T , et al. Docker ecosystem - Vulnerability Analysis[J]. Computer Communications, 2018, 122(JUN.):30-43.

[14] Xiang J , Chen L . [ACM Press the 2nd International Conference - Guiyang, China (2018.03.16-2018.03.19)] Proceedings of the 2nd International Conference on Cryptography, Security and Privacy - ICCSP 2018 - A Method of Docker Container Forensics Based on API[J]. 2018:159-164.

[15] Amith Raj M P , Kumar A , Pai S J , et al. Enhancing security of Docker using Linux hardening techniques[C]// International Conference on Applied & Theoretical Computing & Communication Technology. IEEE, 2016.

[16] Fathollah Bistouni, Mohsen Jahanshahi (2020). Evaluation of Reliability in Component-Based System Using Architecture Topology. Journal of the Institute of Electronics and Computer, 2, 57-71.