Advances in Engineering Research, volume 141
5th International Conference on Mechatronics, Materials, Chemistry and Computer Engineering (ICMMCCE 2017)

Research of Penetration Testing Technology in Docker Environment

Tao Lu[1,a,*], Jie Chen[1,b]

[1] JiangNan Institude of Computing Technology, Wuxi 214083, China

[a] luluao@mail.ustc.edu.cn, [b] jchen@nudt.edu.cn

Keywords: Docker security; Penetration testing; Denial of service; Container escape; Side channel

Abstract: With the increasing use of cloud computing, virtualization has developed rapidly as the key technology. Especially, Docker container technology has become the focus of attention of researchers because of its advantages of easy deployment and high performance. Consequently the research of Docker security has become more and more important. Based on the brief introduction of Docker architecture and components, the security of Docker is analyzed quantitatively, and is compared with the traditional virtualization. The importance of the penetration testing technology in ensuring the security of the container is clarified. Then, the typical penetration testing technology of denial service, container escape and side channel in Docker environment is studied. Finally, the penetration testing process in Docker container environment is analyzed in detail.

1. Introduction

Resource virtualization technology plays a vital role in achieving the on-demand distribution of cloud computing infrastructure services. Traditional hardware-level virtualization such as Vmware, Xen and KVM[1] has reduced the utilization of resources due to the need to virtualize a complete operating system layer. This is a flaw that can not be ignored in large-scale cluster-level applications. To seek more efficient isolation technology alternatives, Docker company launched an open source container project in March 2013，which quickly succeeded and be widely used in various scenes.

Because Docker is so widely used, its security research work is more practical significant. Penetration testing can test whether the final deployed Docker environment can withstand complex stress tests, thus it becomes an essential key technology to ensure the security of the Docker container environment. In this paper, we summarize the various types of penetration testing techniques in Docker environment，focusing on denial of service, container escape and side channel attack. Finally we take a full investigation of the penetration testing process in the Docker environment.

2. Docker Architecture and Components

Docker is designed based on the client server model, the main components can be divided into Docker Client, Docker Daemon, Docker Register three parts. The Docker Client sends a request command to the Docker Daemon via a TCP or Unix socket to get a service. Docker Daemon is a resident process in the background, its primary function is to receive and process Docker Client's requests, manage Docker containers，and request image-related service to Docker Register when needed. The main task of Docker Registry is to manage and store images, providing authentication services to users, and responds to image queries, push or other similar series requisition [2].

Docker Image and Docker Container are also an important part of Docker, Figure 1 shows the relationship between of them. Docker Image is the basic file for Docker Container running when the Docker Container starts, you must first load the Docker Image file through the Docker Daemon to initialize the runtime environment [3]. Docker Container is the core of the Docker, and all Docker modules ultimately serve the management and working of containers.

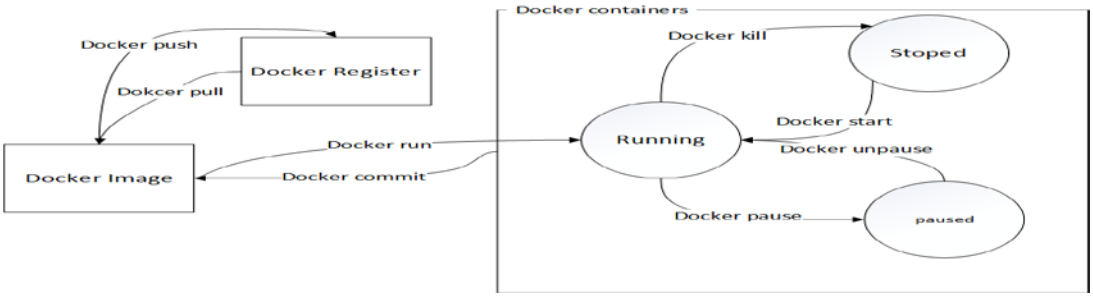

Figure 1 The relationship between Docker Container and Docker Image.

3. Docker Security Analysis

When discussing Docker, it is often compared to traditional virtualization techniques, There has been a lot of research work in this field. Morabito, R., et al[4],Scheepers, M. J. [5],Sharma, P., et al. [6] In their study have concluded that Docker has more performance advantages than traditional virtual techniques. This is determined by the architectures differences between the two, as shown in Figure 2, Docker containers and hosts share the same kernel, do not need to use the Hypervisor layer to re-virtual out of a complete operating system. It is this simplified architecture that determines its natural performance advantages, but also determines its security is weaker than the virtual machine.

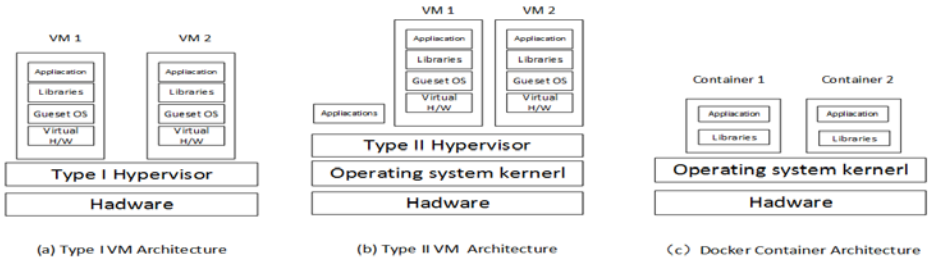

Figure 2 Comparison of Docker containers with traditional virtualization architectures.

In the Docker Container Safety White Paper published by IBM, they made a detailed comparison of the security between docker containers and two types of traditional virtualization[7]. they pointed out that to escape a secure Docker, an attacker in a container would need to find a privilege escalation attack of the shared kernel,such kernel vulnerabilities occur roughly once a year, to escape a type II hypervisor, such as KVM, an attacker must have both a vulnerability in QEMU and a privilege escalation vulnerability in shared kernel, this combination occurs roughly every 2 years, to escape a type I hypervisor, such as Xen,an attacker need to find a vulnerability in the hypervisor, this occurs roughly once every 2 years. So we can define the probability of privilege escalation vulnerability to be $p(k)$,and the probability of type I hypervisor vulnerability is $p(H)$, and the probability of type II vulnerability is $p(l)$.Then define the probability of escape happened in Docker ,Type I hypervisror and Type II hypervisor as EscapeRisk(a)、EscapeRisk(b)、EscapeRisk(c), according to equations (1), (2) and (3) each probability value can be calculated.

$$\text{EscapeRisk(a)}=p(H)=\frac{1}{365*2}=\frac{1}{730} \quad (1)$$

$$\text{EscapeRisk(b)}=p(k)*p(l)=\frac{1}{365*2}=\frac{1}{730} \quad (2)$$

$$\text{EscapeRisk(c)}=p(k)=\frac{1}{365} \quad (3)$$

In Table 1, we give a specific contrast between Docker and traditional virtualization security. It is obvious that the security risk of Docker is higher than that of traditional virtualization technology. The Docker architecture determines the advantages of high resource utilization, as well as security

limitations. Only a better remedy for its security flaws can ensure that it has more advantages, and using security penetration testing technology to strengthen vulnerability detection is a kind of better security matching solution.

Table 1 Comparison of security between Docker and traditional virtualization.

Types	Privilege escalation vulnerability 3	Type I hypervisor vulnerability	Type II hypervisror vulnerability	EscapeRisk
I Hypervisor	☒	☑	☒	$\frac{1}{730}$
II Hypervisor	☑	☒	☑	$\frac{1}{730}$
Docker	☑	☒	☒	$\frac{1}{365}$

4. Penetration Testing Technology in Docker Environment

There are more technical methods for Docker's penetration test, most of the traditional penetration testing techniques, including SQL injection, script execution, phishing attacks, network sniffing, man-in-the-middle attacks, can be applied to Docker environment. This section only focuses on the unique characteristics of Docker environment, focusing on the denial of service, docker container escape, side channel attacks, these three types of typical penetration test technology.

4.1. Denial of Service

Denial of service in the Docker environment will generally run by exhausting the target resource, crashing the target system, and destroying the target hardware, causing the container running on it to fail to get normal service and stop running. The target resources include memory, CPU, storage, network, kernel file handle, random number generator, etc. Chelladhurai gives an instantiation of denying a service by running an MPI program in a container to run out of system memory and CPU computing resources [8]; Combe, T point out that since images are compressed, a specifically crafted image containing a huge file filled with gibberish data (e.g. zeros) would at least fill the host storage device causing denial of service [9];a compromised container with access to /dev/random can disrupt, slow down or possibly entirely halt future cryptographic operations (due to blocking behavior) within the same container host by issuing a large number of reading calls to the device .An attacker can initiate a class of denial of service by exploiting a system vulnerability and collecting system leaks to destroy the operating system and hardware, Some of the leaked information that exists in the collection of pseudo-file systems can provide good helps for denial of service. In their research, Gao, X et analyze the leaked information in the /proc directory, selects the appropriate critical point, and starts a small number of containers at a small cost to launch a power attack on the data center[10];The /sys/firmware/efi/vars files Exposes interfaces for interacting with EFI variables to the container, and the modification of the file can cause damage to the computer that is not repairable[11].

4.2. Docker Container Escape

Docker container escaped will generally use Docker Daemon file parsing vulnerabilities, system kernel privilege escalation vulnerabilities and other means, to achieve the purpose of elevating user rights and break the original isolation mechanism restrictions. According to its use of vulnerability points can be summarized as the use of Docker Daemon file parsing vulnerabilities to achieve the escape; the use of Docker container environment misconfigurations to achieve escape; use of kernel vulnerabilities to achieve escape three cases. Docker Daemon needs to compile the Dockerfile file, parsing image files, if the external input without filtering, when triggered to Docker Daemon loopholes, may cause container escaped.In the early version of the docker, compiling the deformed Dockerfile files and Improper parsing specially constructed soft link file in the images would cause

arbitrary code execution, they all belong to this kind of escape problem [12].Kleindienst described in the article when mounted the /var/run/ directory to the container will lead to container escape [13], and if the CAP_DAC_READ_SEARCH privilege is given to the container by default can cause an arbitrary file access attack, they all belong to mis-configuratation escape problem. Because the Docker container and the host share the same kernel, privilege escalation vulnerabilities in the Linux kernel and driver can be used to achieve container escape. Jian, Z in their paper point out that can though be switching namespaces or through modifying shared memory achieve container escape [14].

4.3. Side Channel Attack

Side channel attacks in a container platform environment generally refer to the detection of side information leaks in adjacent containers, then analysis and speculation the container is running which applications, key information, etc. Zhang et al conducted a more comprehensive study of the side channel attacks under the container cloud platform, which discusses how to implement side channel attacks under the real multi-tenant cloud Paas platform [15]. Schwarz, M et al in their experiment shows how to use the Intel SGX security mechanism vulnerability, recover RSA private keys that in another docker container,because of the attack process directly detects the cache information on the bottom of the system, Docker isolation mechanism can not effectively prevent this attack method [16].

5. Analysis of Penetration Test Process in Docker Environment

Docker ecosystems are complex, and in the face of such a large system, the tester may first consider where to start the penetration test, and ultimately what effect to achieve. This section is based on the tester's perspective, analysis penetration foothold exists in the Docker environment, and the specific penetration test process. Penetration tests first need to find a foothold, and then as a base gradually spread out. According to the structure of Docker environment and external services, the host, container, Docker Daemon, the surrounding environment can be selected as the foothold. Running on the host computer with its own operating system and other server software, which caused a series of security issues, it belongs to the traditional penetration tests work, not in the scope of this article. We will combine Figure 3 to discuss the specific process of three other penetration footholds.

Take the container as a foothold, penetration tester first needs to obtain a container by means of conventional penetration testing or direct leasing. After having a container control should first be carefully analyzed whether the container has privileges, availability of vulnerabilities or resident containers, and then develop different penetration testing strategies based on these. If the container itself has high privileges or has available vulnerabilities, You can try to control the host through a container escaping attack or privilege escalation attack, and then detect the neighboring hosts in the network to find out if there is a host hosting orchestration tools, further control over such a host system is substantially equivalent to taking over the entire container cluster system.if can’t do that directly, you can try to use the container as a springboard, through ARP attacks, MAC flood attacks, sniffing attacks, etc. to reach the control of the adjacent containers, now according to the situation, to detect whether the container can escape to the host, the container is illegal to store sensitive data information, can bypass the system limit policy to initiate denial of service interference co-stationed container, whether through the side channel attack infer the data information in the neighbor container.

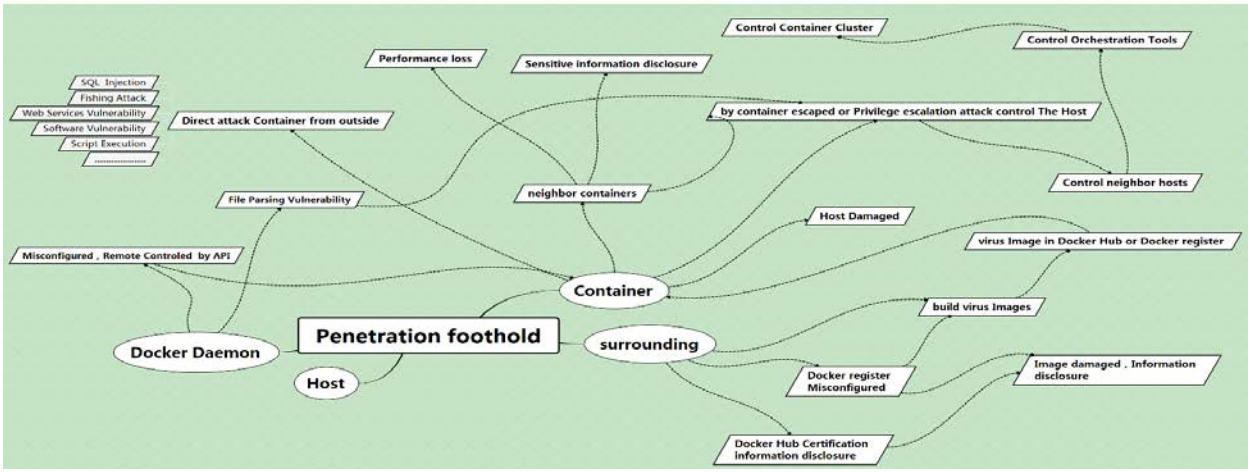

Figure 3 Penetration testing process chart under Docker environment.

Take the Docker Daemon as a foothold, As discussed in section 4.2 above，Docker Daemon may have vulnerabilities when parsing files. The penetration tester should detect whether exist such vulnerabilities can be used to directly control the host. Docker has a well-featured API interface, but there is no mandatory requirement for remote access authentication. Therefore, it should be detected whether there have such misconfigurations maybe cause a series of attack in the system.

Looking for a foothold in the surrounding is also a feasible penetration test scheme, the tester can generate and upload a malicious image to the Docker Hub, to induce the target user to download and use, after an indirect control of the container, launch of a series of subsequent penetration tests. Or directly detect whether the Docker Register is vulnerable, after control Docker Register server, detect whether to modify the user permissions, illegal replacement, delete the images etc. In addition, the tester also needs to detect whether the attacker can successfully initiate a man-in-the-middle attack, hijack or tamper with the images when the host obtains images from the outside.

6. Conclusion

With the increasing popularity of container applications, more and more sensitive applications will migrate to containers. How to enhance container security becomes an unavoidable problem. In view of this situation, this paper makes a quantitative analysis and comparison between Docker and traditional virtualization in the aspect of security, pointing out the security weakness of Docker. We also summarize the typical penetration testing technology detail the specific penetration test process in the Docker container environment. Able by have to test the security is the real security, penetration testing plays an irreplaceable role in ensuring Docker's safety, and systematic research on this is yet to be pursued.

References

[1] Buyya, Rajkumar, Vecchiola, Christian, & Selvi, S Thamarai. (2013). Mastering cloud computing: foundations and applications programming: Newnes.

[2]Jurenka, Vladimir.(2015).Virtualization using Docker Platform.https://edisciplinas.usp.br/ plug infile.php/318402/course/section/93668/thesis_3.pdf

[3] Turnbull, James. (2014). The docker book: Lulu. com.

[4] Morabito, Roberto, Kjallman, Jimmy, & Komu, Miika. (2015). Hypervisors vs. Lightweight Virtualization: A Performance Comparison. Paper presented at the Woc '15: First International Workshop on Container Technologies and Container Clouds. Proceedings of the 2015 IEEE International Conference on Cloud Engineering.

[5] Scheepers, Mathijs Jeroen. (2014). Virtualization and containerization of application infrastructure: A comparison. Paper presented at the 21st Twente Student Conference on IT.

[6] Sharma, Prateek, Chaufournier, Lucas, Shenoy, Prashant, & Tay, Y. C. (2016). Containers and Virtual Machines at Scale: A Comparative Study. Paper presented at the International MIDDLEWARE Conference.

[7] Salman Baset, Stefan Berger, James Bottomley, Canturk Isci,, Nataraj Nagaratnam1, Dimitrios Pendarakis, J. R. Rao,, & Gosia Steinder, Jayashree Ramanatham1. Docker and Container Security White Paper.

[8] Chelladhurai, Jeeva, Chelliah, Pethuru Raj, & Kumar, Sathish Alampalayam. (2016). Securing Docker Containers from Denial of Service (DoS) Attacks. Paper presented at the Services Computing (SCC), 2016 IEEE International Conference on.

[9] Combe, Theo, Martin, Antony, & Di Pietro, Roberto. Containers: Vulnerability Analysis: tech. report, Nokia Bell Labs.

[10] Gao, Xing, Gu, Zhongshu, Kayaalp, Mehmet, Pendarakis, Dimitrios, & Wang, Haining. ContainerLeaks: Emerging Security Threats of Information Leakages in Container Clouds.

[11] Grattafiori, Aaron. (2016). Understanding and hardening Linux containers. Whitepaper, NCC Group.

[12] Bettini, A. (2015). Vulnerability exploitation in docker container environments. FlawCheck, Black Hat Europe.

[13] Kleindienst, Patrick. (16. August 2016). Exploring Docker Security

[14] Jian, Zhiqiang, & Chen, Long. (2017). A Defense Method against Docker Escape Attack. Paper presented at the Proceedings of the 2017 International Conference on Cryptography, Security and Privacy, Wuhan, China.

[15] Zhang, Yinqian, Juels, Ari, Reiter, Michael K, & Ristenpart, Thomas. (2014). Cross-tenant side-channel attacks in PaaS clouds. Paper presented at the Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security.

[16] Schwarz, Michael, Weiser, Samuel, Gruss, Daniel, Maurice, Clémentine, & Mangard, Stefan. (2017). Malware Guard Extension: Using SGX to Conceal Cache Attacks. arXiv preprint arXiv:1702.08719.