PROCEEDINGS OF SCIENCE

RELOCATE: A Container Based Moving Target Defense Approach

Rui Huang[1]
Zhengzhou Information Science and Technology Institute,
Zhengzhou 450001, China
E-mail: xjhr1009@163.com

Hongqi Zhang
Zhengzhou Information Science and Technology Institute,
Zhengzhou 450001, China
E-mail: zhq37922@126.com

Yi Liu
Zhengzhou Information Science and Technology Institute,
Zhengzhou 450001, China
E-mail:liuyi9582@126.com

Shie Zhou
Zhengzhou Information Science and Technology Institute,
Zhengzhou 450001, China
E-mail:942624127@qq.com

In order to cope with border information leakage problem in cloud services, we presented RELOCATE, a moving target defense approach. RELOCATE chose a lightweight operating system virtualization technology named Docker to manage the containers in physical hosts. Docker performs well for tenants' services because of fast initialization and small footprint. Thus, we used Docker clusters to orchestrate the tenants' services. Additionally, we proposed a novel dynamic relocation strategy to mitigate attacks from malicious neighbors by using the moving target defense thought. Lastly, we conducted a simulation experiment in our testbed. Result shows that RELOCATE is efficient and effective to defense border information leakage attacks.

CENet2017
22-23 July 2017
Shanghai, China

[1]Speaker

PoS(CENet2017)008

1.Introduction

The booming up of Cloud computing has gradually changed the whole industrial and academic world of computer science. There are three levels of cloud computing[1]: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS).

Multi-tenant[2], as a key technique of the software as a service (SaaS), is an application program model essentially. In this model, all users and applications can share the same platform of infrastructures and codes. With the development of container techniques, an increasing number of service providers choose container techniques to solve the problems of large scale multi-tenant service deployment. Nevertheless, containers share the same hardware, software, libraries and file system with each other, causing the isolation boundaries not rigorous and exposing a series of safety problems[3]. Such as, border information leakage attacks. The CPU's data caches, e.g.

At present, there are mainly three kinds of approaches to thwart border information leakage attacks[4]. Zhang et al purposed a cryptographic implementation inside virtual machines to maintain safety. Their technique is a more evolved version[5]. Ananta et al presented a self-cleaning intrusion tolerance, but results show that this approach will cause a long downtime[6]. Soo et al purposed NOMAD[7] using VM migration, which is the closest approach to RELOCATE. However, NOMAD is a pure reactive approach and has a high cost.

To solve the problem mentioned above, the proposed approach includes container management architecture and a dynamic relocation strategy. We use Docker[8] to manage containers and conduct dynamic migration to thwart border information leakage attacks. Moving target defense (MTD)[9] is one of the revolutionary technologies in recent years. Inspired from this, dynamic relocation strategy was proposed. Finally, we test RELOACTE in our simulation testbed.

The rest of the paper is organized as follows. In Sect.2, the architecture of RELOCATE is described. A dynamic relocation strategy based on moving target defense thought is designed in Sect.3. Section 4 conducts the simulation experiments and makes an analysis. Section 5 concludes the paper.

2.Architecture of RELOCATE

Our approach named RELOCATE, which is built to be as universal as possible with minimal application customization. Aimed at satisfying the user/system administrator needs with almost no limitations or constraints, RELOCATION arranges containers in different host systematically by means of a lightweight operating system virtualization technology. We choose Docker, a LXC-based container management tool hosted on Linux operating system to sandbox the users' applications. Docker is increasingly popular with developers and service providers due to its small footprint and fast instantiation. With the help of Docker, independent containers are isolated from each other and the underlying host, ensuring the tenants' privacy. The Linux kernel on the host isolates containers applications including processes, network and file systems,

while the cgroups(Control Groups) provide resource isolation including CPU, memory, I/O and network.

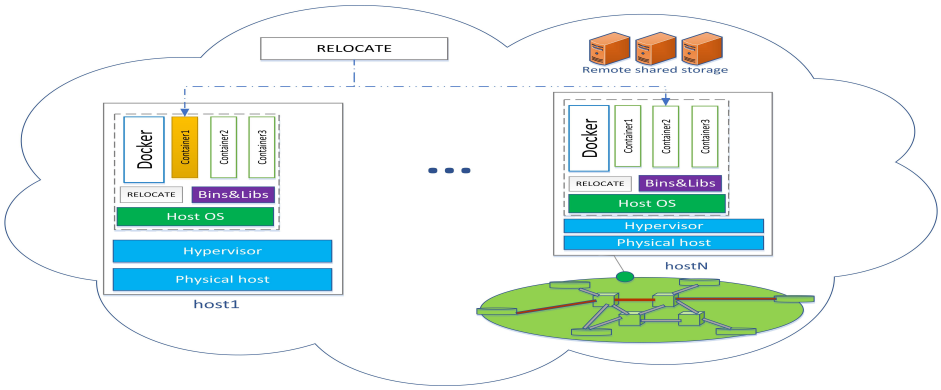


Figure 1 : Architecture of RELOCATE

Figure 1 shows the architecture of RELOCATE. First of all, our approach is based on the cloud services. Every host in the cloud services can either be a real physical machine or a virtual machine. We put Docker cluster into each host with our RELOCATE proxy in it. And another RELOCATE proxy is placed on an independent virtual machine called host0 to cooperate with the inside one.

The RELOCATE process starts by the time clock. Since initiating the container, operation time is dynamically monitored. Once the operation time exceeds the specific threshold, the time clock alerts. Then, the checkpoint and restore module is triggered. There are two main function of this key module: one is the memory dump of containers, and another is network relocation. On finishing these two procedures, dynamic migration module is invoked. The entire process occurs in matter of milliseconds with almost no effect on the enclosed applications.

The entire RELOCATE architecture includes container instantiation, container networking functions and checkpoint and dynamic migration modules. We will illustrate the first two functions in the first place, and the task of modules will be described in next section.

A: container instantiation

First of all, we are supposed to describe the working procedure of Docker. Since Docker is a lightweight operating system virtualization technology, every operation is executed through images. If the local host has the needed image, then the container can be started according to it. Otherwise, you can download from the Docker hub. After that, mount the file system and assign the IP address. Then, the container can be used to run the applications. In our condition, tenants will instantiate containers either manually or using an automatic script. The container will hold the tenants' application, and all the needed files for that application to run.

B: container networking

It is extremely similar between container network technology and virtual machine network technology. Since Linux containers are a software construct that can host an application and its dependencies as an isolated process on a Linux kernel. It allows sharing the kernel with other containers. In case of default installation, Docker will set up a virtual bridge called docker0 in the host machine. This virtual bridge has a private network address and its subnet. And most of the time, docker0's address is 172.17.42.1. All the containers will be linked to the virtual bridge

and assigned an IP address in the subnet. As a result, it allows pass packets back and forth between containers on the same host. Generally, there is only one NAT interface to access external network.

3.Dynamic relocation strategy

RELOCATE mainly consists of three modules: checkpoint and restore module, network relocation module and dynamic migration module respectively. Figure 2 shows the process of dynamic relocation strategy. There are two main aspects of the problem need to pay great attention to:

1- When to relocate?

Docker Daemon dynamically set up containers for each tenant according to the images in repository and the dockerfile. The container is responsible for the operation of the application, including operating system, user files and data. Nevertheless, during the execution of an application, the start time is about one-tenth of the total time, and the rest of the time is the application's running time, which causes the application to be exposed to the attacker for a long time. As a result, the vulnerability is easily detected. Great attentions should be paid to this issue. We are inspired from the offensive and defensive rules of badminton, that is, immediately return to the midfield and empty state after the defense or attack. Return to the start point so that the adversary's attack cannot be accumulated. In our strategy, under the premise of continuous service, RELOCATE switches the containers constantly. It ensures the safety of large scale multi-tenant service deployment by shorting the life cycle of containers. We set up a time clock and use Poisson distribution to calculate the time, regarding each process k as an independent event of Poisson distribution. The symbol λ is the probability distribution index, it is always a constant. We define the symbol T as the threshold. Once the time exceed the threshold T, the time clock will alert.

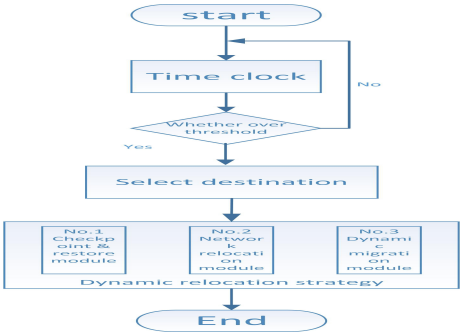
$$P(X=k)=\lambda^{k}/k!\,e^{(-\lambda)} \tag{3.1}$$

$$P(X=k)>T \tag{3.2}$$

2- Which one (host) should be relocated in?

Prior to starting the dynamic migration module, RELOCATE should select a suitable destination. With the purpose of not reusing the same vulnerability, RELOCATE needs to select a heterogeneity scale host. Heterogeneity scale is decided by host configurations C, networking N and services S, e.g. we define heterogeneity entropy Δ to measure the difference between the hosts. Different indicators hold different weights. The higher Δ is, the more likely for RELOCATE to select. In the mainwhile, it is regarded as a safe, reliable and long term benefit relocation.

$$\Delta=w_c(C_d-C_s)+w_n(N_d-N_s)+w_s(S_d-S_s) \tag{3.3}$$



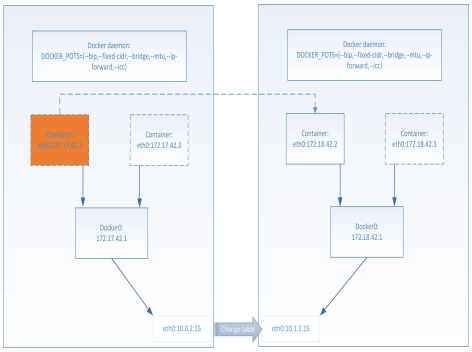


Figure 2 : Process of RELOCATE **Figure 3 :** Network relocation procedure

3.1 Checkpoint and restore module

Since the container is a lightweight encapsulation of the application, including many important information, such as process and memory information, we should dump the memory before migrating. We leverage the encapsulation state of the application and used CRIU to dump the container memory into persistent set of files that are easy to share and recover.

CRIU is an open source tool to checkpoint and restore running tasks in user space[10]. CRIU momentarily freezes the running (container) runC process and all its sub-processes (user apps) and checkpoint it to a collection of image files that can be used to restore the container to the next state later. The dumped images are stored in persistent storage. In order to minimize the application response time, containers will use the host memory to operate between these dump events.

RELOCATE uses a remote shared storage as a container repository to store runC containers, ensuring fast instantiation, quick recovery and easy migration. Running the container from a remote storage gives easy access to initiate such containers. RELOCATE sets up a remote repository to host the base image of containers, which massively reduces the time needed to move all the files between hosts. The only files that have to be synchronized between the source and destination host are the memory dump which are small and easy to be relocated.

3.2 Network relocation module

After the checkpoint and restore procedure, the state of original container has been stored in persistent set of files, including processes, memory and so on. It should be noted that CRIU can only restore the state memory but network relocation issue is ignored. Assuming that a tenant under docker0's subnet is downloading a file from out net, simply checkpoint and restore will breakdown the download process, which cannot guarantee services for the tenant. Furthermore, if the malicious tenant neighbors the target, having stolen some vital information of the container, such as IP address. Then, the attacker may launch Dos to destroy the target. Great attentions should be paid to this issue.

In our approach, two procedures were purposed to solve the problem. Figure 3 illustrates the procedure. Firstly, RELOCATE kills the process on the original host and makes an ARP update to change the MAC/IP assignment of the old container. Then, the new docker0 in destination host will assign a new IP to the destination container. New IP address will make the

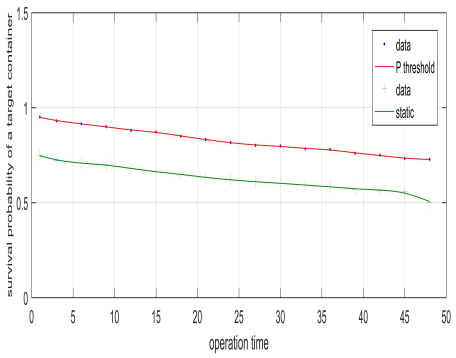
attacker's accumulated knowledge invalid. Secondly, in order to maintain the service of tenants, we modify the host NAT interface's destination address to the new host NAT interface, ensuring data flow not lost and achieving a virtual IP hopping from the source to the destination.

3.3 Dynamic migration module

For RELOCAION working, the participated host must run a Linux operating system customized kernel to enable some features. After the two modules, it is time to migrate. The destination container will recover the application from the remote shared storage and rebuild the network connection. The entire process occurs in matter of milliseconds, guaranteeing almost zero downtime to the tenants.

4.Simulation experiments and analysis

We built a simulator to evaluate the effectiveness of RELOCATE. Additionally, we conducted our experiments on our local cloud with 30 virtual machines (hosts) in it. Both the virtual machines were equipped with Linux CentOS 6.5 operating system. The experiment included two parts: time migration and space migration. We regard the attack succeed only if the attacker manages to keep the target sharing the same host with his containers for the entire time of attack. We assumed that we cannot identify the attacker containers and the main goal was to minimize the chance of sharing the same host with any untrusted container. For time migration: we tested static and probability threshold conditions. Figure 4 shows that more than 87% of the border information leakage attacks can be resisted by RELOCATE. For space migration: we tested static, high heterogeneity scale and low heterogeneity scale conditions. Figure 5 shows that RELOCATE can improve approximately 29% survival probability of a target container compared with the static condition. Both the results indicated that the effectiveness of our purposed approach.



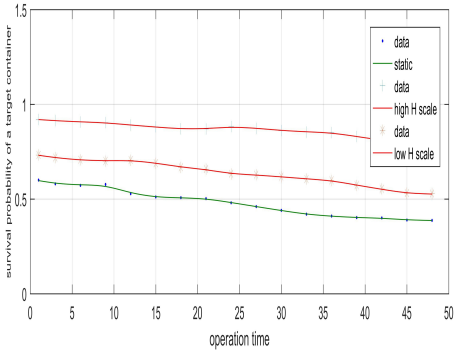


Figure 4 : Time migration

Figure 5 : Space migration

5.Conclusion

In this paper, we introduced RELOCATE, a container based moving target defense approach to thwart border information leakage attacks in cloud service. We purposed a novel architecture using a lightweight operating system virtualization technology named Docker. And

we designed a dynamic relocation strategy to mitigate the attacks. Simulation experiment showed that RELOCATE can effectively avoid border information leakage attacks with almost no downtime to tenants. Our future work is to evaluate RELOCATE in a real condition and make further efforts to quantify the heterogeneity scale.

References

[1] Nist S P. A NIST definition of Cloud computing[J]. Communications of the ACM, 2011, 53(6):50-50.

[2] Shen Z, Subbiah S, Gu X, et al. CloudScale: elastic resource scaling for multi-tenant cloud systems[C]// ACM Symposium on Cloud Computing. ACM, 2011:5.

[3] https://www.dockerbook.com/

[4] Bangalore, Anantha K, and A. K. Sood. "Securing Web Servers Using Self Cleansing Intrusion Tolerance (SCIT)." International Conference on Dependability IEEE Xplore, 2009:60-65.

[5] Wu, Zhenyu, Z. Xu, and H. Wang. "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud." Usenix Security Symposium (2013).

[6] Bates, Adam, et al. "Detecting co-residency with active traffic analysis techniques." Proceedings of the 2012 ACM Workshop on Cloud computing security workshop ACM, 2012:1-12.

[7] Zhang, Yinqian, et al. "HomeAlone: Co-residency Detection in the Cloud via Side-Channel Analysis." IEEE Symposium on Security and Privacy IEEE Computer Society, 2011:313-328.

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

[9] Jajodia S, et al. Moving Target Defense[J]. Advances in Information Security, 2011, 54:99-108.

[10] http://criu.org/Checkpoint/Restore