DOI: http://dx.doi.org/10.26483/ijarcs.v9i2.5733

ISSN No. 0976-5697

Volume 9, No. 2, March-April 2018

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

AN APPROACH TO IMPROVE ISOLATION AND SECURITY IN CONTAINER BASED CLOUD SYSTEMS.

Salini Suresh
Department of Computer science
Seshadripuram Academy of Business Studies
Bangalore, India

Manjunatha Rao
Department of MCA
Dr. Ambedkar Institute of Technology
Bangalore, India

Abstract: Container-based virtualization is a lightweight virtual hosting environment that provides application isolation with less overhead. Containerization enables the creation of isolated, multiple user-space instances and effectual consumption of resources and rapid provisioning. However, container-based virtual environments have a weak isolation due to a shared kernel. This makes the entire system vulnerable to security attacks. This paper investigates various security issues in container based systems and proposes a solution for securing the container using a novel access control model. We also conducted stress test using benchmarking tool to evaluate the enhanced isolation using our proposed model.

Keywords: cloud, container, security, isolation, authorisation, access control.

I. INTRODUCTION

Virtualization enables resource sharing using emulation of resources. The hypervisor-based virtualization provides sufficient isolation between guest virtual machines but imposes significant overheads. On the other hand, in container-based virtualization, the host operating system is virtualized to aid multiple guest containers run on top of host operating system without installing the added kernel for the guest containers, which substantially reduces the performance overhead [1].

Kernel namespaces and control groups are deployed to realize isolation between guests and host operating system in Linux kernels. Namespaces offer resource isolation, network isolation while cgroups bring about the resources, process control and depict the configuration of a network. The multiple processes added to cgroup shares all the resources allocated to that group [2]. LXC has also taken on of Linux Kernel features like chroot, Process identifier (PID) and permit users to build and manage applications using APIs.

Even though namespaces and Linux kernel feature perk up container security, there exist various security issues due to a shared kernel and weak isolation. In this paper, we investigate the security risks associated with a container based virtual environment.

The major contributions of this paper are the following:

- We identify the security concerns in container based virtual environments.
- We review the mechanisms to mitigate the identified security issues.
- Propose a model to enhance the isolation and security of the cloud container-based environment.
- Benchmarking the proposed model and evaluate the performance.

The rest of the paper is organized as follows:

In Section II we have discussed security concerns in cloud container based environments. Section III details the existing security mechanisms that are deployed to address the challenges in container security. Section IV explains the proposed approach. Section V explains the access control process flow of the security module. In section VI evaluation of proposed system and the results are discussed.

II. SECURITY VULNERABILITIES

Containers suffer from weak isolation owing to sharing of the system kernel. Any malicious code can get proliferated to other containers and underlying operating system. A malicious container on the host can compromise the security of other containers and thus the entire system.

A user with root privileges on the container can mount and access any directory from the host even the root directory [3] which exposes the host to a threat called root breakout. A process that breaks out of the guest container has same access privileges on the host including root privileges. Host break in attack turns out on the allocated container when the host root gains access to containers and modifies it [4]. A user with uid 0 will be root on the host with full root access privileges on the host. This can lead to privilege escalation attacks where a user gets privileges like root users.

The multi-tenant container cloud-based systems where multiple containers share the host resources are vulnerable to threats like information leakage [5]. Adversaries can take advantage of these information leakage channels and instigate attack that affects the confidentiality and integrity of the entire system [6].

The container daemon is easily accessible from a web interface which makes the applications in the container it prone to security attacks. Malicious applications in the container to tire out the system resources to an extent those other applications are unable to use them, like in the case of threats like Denial of service (DOS) [7]. Container lacks an additional layer of isolation provided by the hypervisor which makes it effortless for attackers to launch into the host machine. Cloud container based environments that lack strong access control policies are susceptible to cross site scripting [8] attacks that let the attacker instill client side script into web pages.

III. SECURITY APPROACHES

A. ACCESS CONTROL MECHANISMS

The authentication and authorization of users to resources is a major challenge in a multiuser environment. Numerous security issues caused due to multi-tenancy can be addressed by strong access control methods [9]. Some of the traditional access control methods used in cloud systems are discussed below.

1) Discretionary access control (DAC)[10]

DAC enables resource owner to set the access rights for the users and restricts the right to use of resource according to the identity of the user. The access privileges of each user are specified using Access control list (ACL), and these owners defined policies are applied to the files and directories by the administrator. DAC is the default access control method in UNIX systems that apply security attributes to the subjects and objects, allows controlling access to objects and setting rights at user's discretion.

2) Role-Based Access Control (RBAC) [11]

RBAC is nondiscretionary access control method that allows access to the resources based on the subject's role in an organization. User's RWE rights on the container objects dependent on his role and does not have the command over the roles assigned. A core RBAC model assigns roles to users with specific rights. The user can perform operations on the objects according to permissions assigned. ACL was inefficient in cloud environments with a large number of users. Hybrid approaches using RBAC and ACL are used for secured authorization.

The main shortcoming of DAC and RBAC models is that these models implement the access control checks through the application interfaces, which are easy to be evaded in web-based systems [12].

3) Mandatory Access Control (MAC)[13]

Mandatory Access Control enables administrators to define and implement access control policies for the entire system that cannot be bypassed by the subjects.MAC policies override the DAC file and directory privileges. The mandatory rules of "no read up "and "no write down "are practiced in MAC to ensure security. The most prevalent MAC technologies for Linux are SELinux and AppArmor and are realized using Linux Security Modules (LSM) framework.

a) SELinux [14]

SELinux administers policy-based security controls that define activities permitted for a process on a system irrespective of DAC permissions. Access policies are set as labels for entire system components which include files, directories, process and other container objects. It provides a strong level of isolation for the containers.

b) AppArmor[15]

AppArmor is a Linux security module (LSM) implementation and is a Mandatory Access Control (MAC) system that achieves fine-grained access control system to protect from attacks by allowing an application to access the objects specified in AppArmor Profile. It follows a file path based approach in contrast to SELinux which follows label based method. The enforcement mode of AppArmor enforces policies on an application and reports any policy violations. The enforcement mode of AppArmor restricts a container from retrieving important file system on the host in case a process in the container is compromised.

c) Information flow control (IFC)

Information flow control is a MAC based approach to control and make safe the data propagation.IFC employs a controlled and safe information flow by attaching labels with data and with users who desire to access the data. IFC model defines secrecy and integrity labels that can be associated with system entities. Data flow is endorsed if the security label of the source is a subset of the label of the receiver [16].

IV. PROPOSED APPROACH

We propose a security module for improving the isolation and security of container-based cloud systems. Our model relies on employing stronger access control mechanism combined with process id (PID) isolation to address the security challenges caused due to a shared kernel. The architectural outline of our approach is as shown in Figure 1.

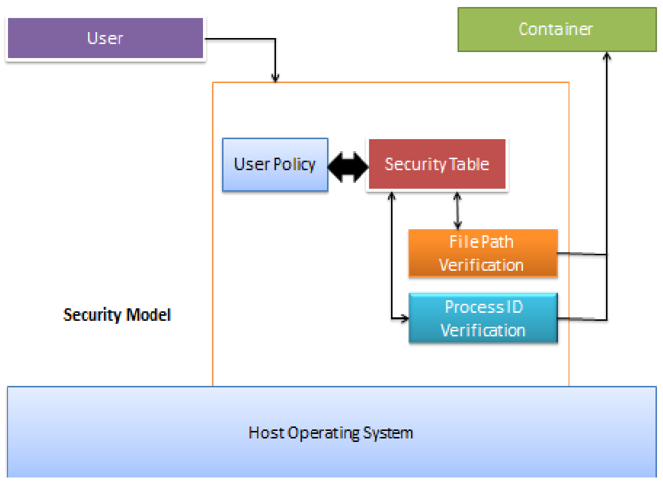

Figure 1: Schematic diagram of proposed approach

An authenticated user who login to the system will have a defined user policy. In the security module file path of each container object is mapped against a user and each process id in turn mapped against user security table. The security table maintains an index of all file path and process id. It also maintains a log of all the container activities in the system.
The two-way authorization mechanism verifies object file path against the user policy and hence ensures that only authorized users who have an entry in file path access the container. Even the host root is denied access to the container as it does not have a file path entry in security table. Each Process id is mapped to the user so that any user without a mapping to process id will not permit access to the container objects.

V. ACCESS CONTROL PROCESS IN SECURITY MODULE

- Step1: User authentication on credential verification
- Step2: User policy applied.
- Step3: Object file path set in the security module.
- Step4: Set the process id and update the security table.

- Step5: if user policy equals the security table entry access is permitted to the container.

VI. EVALUATION OF CONTAINER ISOLATION

In order to evaluate our proposed security module, we created a system prototype with Ubuntu 14.4, open stack and container on LXD. We conducted fork bomb test using benchmarking tool to evaluate the enhanced isolation using our proposed model. Fork bomb test creates the denial of service (DOS) and exhausts the resources by looping that creates new child processes. Fork bomb operates on CPU cycles and saturates the system. In our proposed approach as we have deployed process id isolation and file path verification, a user is permitted to run process inside the container as per the user policy and this ensures that unauthorized processes are not created.

The analysis of the results from fork bomb test illustrates that with proposed system an improvement of 99% was achieved in reply time, 33% improvement in Net I/O, 24% increase in connection rate and 26% increase in test duration. Hence the proposed approach increases the container isolation and improves the security in container based cloud systems.

Graphical representation of the performance benchmarking between a native container and container secured with our proposed approach is as shown in Figure 2.

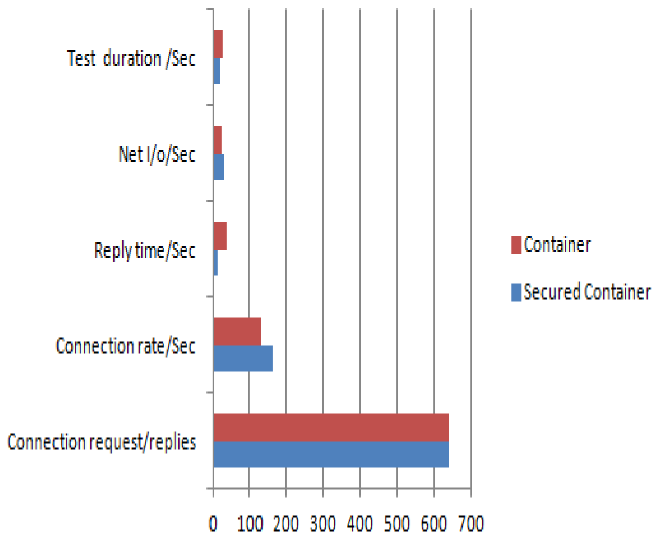

Figure 2: Performce comparison

VII. CONCLUSION AND FUTURE WORK

The dynamic, ad-hoc nature of cloud computing demands much more than traditional security. We have discussed security concerns in cloud container based environments, security mechanisms and proposed a novel approach for a secured access control. Our future work will focus on integrating flexible access control and multi policy to the proposed model.

VIII. REFERENCES

[1] S. Soltesz, H. P¨otzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, “Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors,” in ACM SIGOPS c Systems Review, vol. 41, no. 3.ACM, 2007, pp. 275–287.

[2] https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt

[3] Docker Security”. http://docs.docker.com /articles/security.

[4] PROTECT: Container Process Isolation Using System Call Interception, Thu Yein Win; Fung Po Tso; Quentin Mair; Huaglory Tianfield,2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC),Year: 2017,Pages: 191 – 196.

[5] Securing the infrastructure and the workloads of linux containers,Massimiliano Mattetti; Alexandra Shulman-Peleg; Yair Allouche; Antonio Corradi; Shlomi Dolev; Luca Foschini,2015 IEEE Conference on Communications and Network Security (CNS),Year: 2015,Pages: 559 - 567.

[6] ContainerLeaks: Emerging Security Threats of Information Leakages in Container Clouds Xing Gao; Zhongshu Gu; Mehmet Kayaalp; Dimitrios Pendarakis; Haining Wang,2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN),Year: 2017,Pages: 237 – 248.

[7] Securing Cloud Containers Using Quantum Networking Ch annels,Brian Kelley; John J. Prevost; Paul Rad; Aqsa Fatima,2016 IEEE International Conference on Smart Cloud (SmartCloud),Year: 2016,Pages: 103 – 111.

[8] Secure Cloud Container: Runtime Behavior Monitoring Usi ng MostPrivileged Container (MPC),Vivek Vijay Sarkale; Paul Rad; Wonjun Lee 2017 IEEE 4th International Conference on Cyber Security and CloudComputing (CSCloud), Year: 2017 Pages: 351 - 356

[9] A Secure and Reputation Based Recommendation Framework for Cloud Services. M. Thangapandiyan, P. M. Rubesh Anand 2016 IEEE International Conference on Computational Intelligence and Computing Research.

[10] Sandhu, R. S., & Samarati, P. (1994). Access control: principle and practice. Communications Magazine, IEEE, 32(9), 40-48.

[11] D. Ferraiolo, R. Kuhn, “Role-Based Access Controls,” In: 15th NISTNCSC National Computer Security Conference, Bultimore Maryland,USA,1992,pp.554--563.

[12] M. Dalton, C. Kozyrakis, and N. Zeldovich, “Nemesis: preventing authentication & access control vulnerabilities in web applications,”in *2009 USENIX Security Symposium*.

[13] Lindqvist, H. (2006). Mandatory access control. Master's Thesis in Computing Science, Umea University, Department of Computing Science, SE-901, 87.

[14] Wright, C., Cowan, C., Smalley, S., Morris, J., & Kroah-Hartman, G.(2002, August). Linux Security Modules: General Security Support for the Linux Kernel. In USENIX Security Symposium (Vol. 2, pages 1-14).

[15] https://help.ubuntu.com/lts/serverguide/apparmor.html

[16] A. C. Myers and B. Liskov, “A decentralized model for information flow control,” in Proc. 17th Symp. Oper. Syst. Principles, 1997,pp. 129–142.