International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 02 | Feb 2020 www.irjet.net p-ISSN: 2395-0072

A Comprehensive Review on Security Issues and Challenges in Lightweight Container Communication

Sathya Priya A[1], Dr. Shashikumar D.R.[2]

[1]Asst. Professor, Dept. of CSE, Cambridge Institute of Technology, Bangalore.
[2]HOD & Professor, Dept. of CSE, Cambridge Institute of Technology, Bangalore.

Abstract:- *Virtualization is a technique that allows more than one Virtual Machines (VMs) to reside on a single physical machine. It is known, that due to the multiple virtual machines in a single physical machine affects system performance, and it largely vulnerable to many networked attacks than conventional physical machines. To overcome such an issue, the evolving lightweight cloud technology, such as containers, is gaining wide grip in IT. Containers are the lightweight alternatives for the virtual machines and that allows users to deploy applications in any environment faster and easier. Operating system level container-based virtualization is able to provide more efficient and lightweight virtual environment to deploy microservices, but not without security distresses. Previous surveys are focused on the different types of attacks generally. In this paper, the focus is on the four different use cases and its issues and challenges are explored. Containers leverages on Linux kernel security features such as kernel namespaces and control groups (cgroups). Mostly, container based virtualization subjects to the attacker who can perform different types of attacks, such as Denial-of-Service (DoS), privilege escalation, poisoned image and kernel exploits. Hence, we need to have research on security enhancement for containers to deal with such issues.*

Keywords: Containers, Lightweight virtualization, Docker.

1. INTRODUCTION

Containers do operating system virtualization like hardware virtualization done through hypervisors. Containers can encapsulate the data as lightweight. Wraps up a portion of software in a complete file system that contains everything that needs to run such as code, runtime, system tools, libraries etc., they share the OS kernel and bins/libs where needed, otherwise each of them operate during a self-contained environment. Dockers, LXC are some of the most popular implementations of containers today [1]. Cloud industry started adapting container technology to deploy cloud services and microservices. OS-level virtualization may be a method by which the OS kernel allows multiple isolated user-level processes to share the resources of the system. It provides the illusion that a gaggle of processes has its own set of kernel and hardware resources. Containers are built on top of this OS-level virtualization. These virtual environments are often referred to as containers [2]. The virtualized containers are similar to the ordinary process which runs on top of the kernel sharing the host machine resource. Containers provide isolated environments with the required resources to execute applications and these resources can be either shared with the host system or the one which is already installed in the container. Containers aim at solving a long-standing issue within the software development and deployment world, portability [3]. With these, users can easily create customized environments perfectly tailored to fit the needs of their software products. Docker containers are built on top of resource control groups (cgroups) and namespaces. Resource control groups virtualize hardware between processes. They also limit and isolate the resource usage (CPU, memory, disk I/O, and network). Namespaces allow to have its own identify for the container in terms of hostname, process IDs, network, file systems, user/groups, and inter-process communication. These two functions together allow the creation of containers on a single host OS that are isolated from each other and can be customized for the needs of the applications running inside them [4]. Fig.1 shows the difference between the virtualization and Containerization.

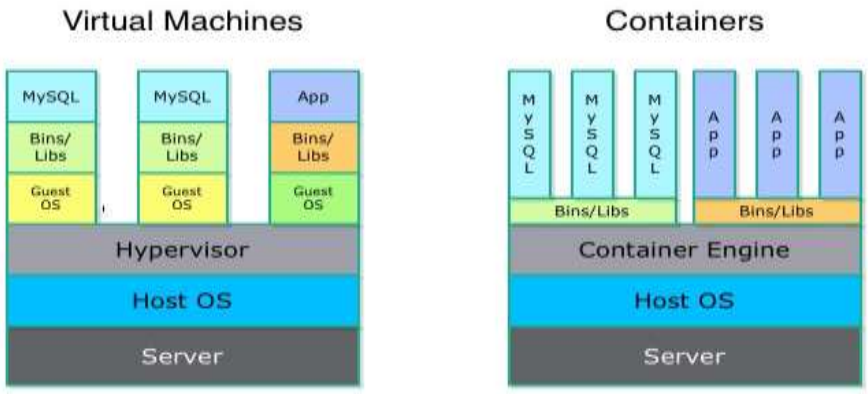

Fig.1: Virtual Machine Vs Container [18].

2. BACKGROUND

In this section, we present backgrounds for containers, few container securities issues and microservices architecture.

2.1 Containers

Containers are referred using different names in this survey as OS level virtualization, Docker, lightweight virtualization and lightweight communication [5].
Containers are more effective and efficient when compared with the virtual machines for majorly two reasons, first they shares the same OS kernel where virtual machines need a separate one. Second, as containers are lightweight they can be started and stopped very easily whereas virtual machines want more time to do so. Containers are more efficient for running microservices, which made the effort easier and faster [6].
Containers have its own benefits but in the other part it needs to face many challenges like

1. Exploitation of the kernel: For the process which is being executed inside a VM is hard to damage the host kernel when compared to the container doing that damage. Because the process which is running inside the VM should first break the VM kernel and then the hypervisor layer to tough the host kernel. These two levels of protection are there for virtual machine but for containers, they share the host kernel, so it is very easy to exploit the host kernel. Even a very small level of liability in the application can break the kernel [7].
2. Denial of Service Attacks: The container assumes that, there is only its own process is being executed in the host machine. Even when we execute the ps command it lists only the information about process which is inside its container not about other process. However, if the container breaks out, then it will start accessing the resources like CPU cycles, memory, user IDs (UID) etc. of the host system. Once it acquires the control of all resources it will not allow the other parts to use that resources [7].
3. Container Breakouts: Up until recently, Docker didn't have the concept of providing username for containers, which is being the major security concern. If process have all privileges to access the host system, if that kind of system is broken ie breaks out of the container then it can access the root system also with the same privileges [8]. This may lead to privilege escalation attack i.e., when one user gains system rights of another user.
4. Poisoned Images: Docker hub, is the official registry for Docker containers, which have the privileges to store millions of images and it can be downloaded by any Docker user for his deployment. These images do not have any identification for trustworthiness [8]. Means, it may be from a hacker too, that is a hacker can add a malicious code in the image through which he can gain the access to the entire system. The images may be outdated and may contain much vulnerability.
5. Compromising secrets: There may be certain applications that need a database access for the execution of it. Most of those services may require API keys and database passwords for connecting to the required database access. If an attacker gains access to those secrets, using these secrets he can gain the complete access of the service too. So, these passwords and keys must be kept in a secure way [8].

2.2 Microservice Architecture

A monolithic application is one in which the parts of the software are strongly coupled and cannot be executed separately [9]. Even to run monolithic application in container is better to use microservice architecture. Actually, microservice applications contain loosely coupled components which can run independently. Microservices allow the developers to implement the new technologies and adopt it in development of software [8]. For example, online shopping application can have different services like payment service, catalog service etc., which are implemented as microservice. Fig.2 shows the microservice architecture and its different services which it can handle [10].

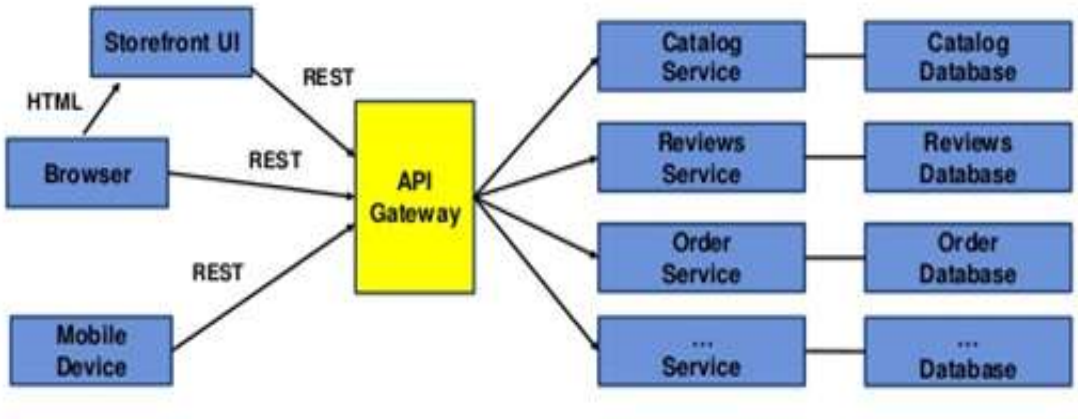

Fig.2: Microservice Architecture [18].

2.3 Literature Review

In this review, the considered is majorly on the four general use cases for host-container level. The use cases are

i) Defending containers from semi-honest or malicious host

ii)Protecting containers from the applications

iii) Protect the host from containers

iv) inter-container protection.

3. MODEL

The main focus is on the host-container level, these use cases are used to pour more light on the host and the container which is a primary focus on container technologies.

3.1 Use Case 1: Defending container from semi-honest or malicious host

When the containers run on the untrusted host, leads to integrity breach of containers and its applications. A semi-honest host can gather the information about the containers because it handles its network devices, storage and processor. So, it is better to avoid using containers in Container as a Service from those service providers [3]. There can be many numbers of active and passive attacks can be done in container using this malicious host. Active attacks are more dangerous because it can change the behavior of the application [11]. As there is a possibility of all types of attack because of malicious host, we didn't represent it in the form of table.

3.2 Use Case 2: Protecting container from applications

In this use case the main focus is on the intra container attack. Actually, the applications can't break the access control polices set for it, but few applications need root privileges [12]. If it acquires the root privileges it is easy it to gain the control of container manager and then target the host system in turn to the other containers. There are many possible attacks through this. Table 1 describe about the different types of attack which are possible on the use case 2.

Table 1: Possible threats for use case 2 [13].

Sl. No.	Threat	Attacks - Possible	State of attack	Research Possibility/Direction
1.	Untrusted image -usage	Denial of service, backdoor threats etc.	Untrusted images can have preinstalled backdoor inside them.	Can use only images from trusted registries and it should be verified.
2.	Application liabilities	Denial of service (DoS)	A vulnerable application can have DoS attack which target on the availability of container resources.	Root File system should be read only. Need to scan the application frequently by tool. Application can be given least privileges.
3.	Liability within container runtime	Container runtime escape attack and Privilege escalation attack.	When Docker provide privileges to application for modifying the capabilities.	Container runtime should be monitored and updated periodically.
4.	Embedded secrets	Privacy issues, tampering and information disclosure.	Database connection parameters are stored in the containers in the form of clear text or in simple encryption.	Secrets can be stored outside the image and accessed when it is needed.
5.	Embedded malware	Trojan, worm, virus, ransomware etc.	Ransomware gaining access to the containers.	Monitor the process with anti-malware. Running only the trusted applications.
6.	Configuration defects in images	Unauthorized access and network based intrusion.	Running applications with root privileges access the control of containers. Enabling remote access, if it is not configured properly.	Running applications on least privileges and containers can be managed using runtime APIs.
7.	Image Liabilities	Remote code execution	Executing code through remote connections will affect the containers.	Vulnerability scanning should be done periodically.

3.3 Use Case 3: Protect host from containers

In this use case we assume that atleast one of the containers is malicious or semi honest. If that container gains access to the host, then it completely breaches the integrity of the host system. Very important issue for the host is to prevent the resource drainage attacks on the software [3]. Table 2 describe about the different types of attack which are possible on the use case 3.

Table 2: Possible threats for use case 3 [14].

Sl. No.	Threat	Attacks - Possible	State of attack	Research Possibility/Direction
1.	Resource accountability	Denial of Service attack	Container may consume most of the host	Capabilities are used to control the number of resources to be used by

			resources and containers.	each container.
2.	Host OS kernel sharing	Escape attack on container	Remote code execution	Periodic vulnerability scanning for the images, applications, container runtime, namespace and capabilities are important to control [15].
3.	Host filesystem	Tampering	Mounting a local file system on the host allows a container with sufficient permission to cause data tempering [16].	Containers should run with minimal set of permissions and file changes in containers should be specific.

3.3 Use Case 4: Inter-container protection

When the containers are inside the host, an attacker can take control over the application and in turn it acquires the access to the host resources and can perform many types of attacks [8]. Table 3 describe about the different types of attack which are possible on the use case 4.

Table 3: Possible threats for use case 4 [17], [18].

Sl. No.	Threat	Attacks - Possible	State of attack	Research Possibility/Direction
1.	Container runtime insecurity	DoS and remote code execution	Remote code execution vulnerabilities and configuration error in runtime.	Container runtime should be monitored and periodically updated.
2.	Inter-container traffic	ARP spoofing, MAC flooding and DoS	If traffic among containers are poorly separated may lead to man-in-the-middle attack and flood the network.	Containers should not be able to communicate unless necessary.
3.	Application liabilities	Denial of service (DoS)	A vulnerable application can have DoS attack which target on the availability of container resources.	Need to scan the application frequently by tool.
4.	Untrusted image -usage	Denial of service, backdoor threats etc.	Untrusted images can have preinstalled backdoor inside them.	Can use only images from trusted registries and it should be verified.
5.	Unbounded network access	Port scanning	It targets the vulnerable containers to access the other containers.	Containers should be separated according to the sensitivity level.

4. CONCLUSION

Containers are important for the future cloud computing. Microservices and containers are strongly connected, and it is the easier way to deploy the microservices. However, one of the primary problems for adopting containers for their implementations is the security issues. In this review we mainly focus on these four use cases and its issues. These use cases can be solved either by software based solution or hardware based solutions. The tables are comprised with the attacks and

possible research direction for the future requirements. These challenges may promote fundamental research in these areas and generate huge economic values in future.

REFRENCES:

[1] M. Httermann, DevOps for Developers. New York, NY, USA: Apress, 2012.

[2] C. Tozzi. (Jan. 2017). What Do Containers have to Do With DevOp anyways?. http:// containerjournal.com /2017/01/11

[3] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, ''Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers,'' IEEE Wireless Commun., vol. 24, no. 3, pp. 48–56, Jun. 2017.

[4] H. Khazaei, H. Bannazadeh, and A. Leon-Garcia, ''SAVI-IoT: A selfmanaging containerized IoT platform,'' in Proc. IEEE 5th Int. Conf. Future Internet Things Cloud (FiCloud), Aug. 2017, pp. 227–234.

[5]A.Celesti,D.Mulfari,M.Fazio,M.Villari,andA.Puliafito,''Exploring container virtualization in IoT clouds,'' in Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP), May 2016, pp. 1–6.

[6] R. Morabito, R. Petrolo, V. Loscrì, N. Mitton, G. Ruggeri, and A. Molinaro, ''Lightweight virtualization as enabling technology for future smart cars,'' in Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM), May 2017, pp. 1238–1245.

[7] R. Morabito, I. Farris, A. Iera, and T. Taleb, ''Evaluating performance of containerized IoT services for clustered devices at the network edge,'' IEEE Internet Things J., vol. 4, no. 4, pp. 1019–1030, Aug. 2017.

[8]P.Jamshidi,C.Pahl,N.C.Mendonça,J.Lewis,andS.Tilkov,''Microservices: The journey so far and challenges ahead,'' IEEE Softw., vol. 35, no. 3, pp. 24–35, May/Jun. 2018.

[9]B.Golden.(2016).3ReasonsWhyYouShouldAlwaysRunMicroservicesAppsinContainers.Accessed:Nov.11,2017.

[10] M. Souppaya, J. Morello, and K. Scarfone, Application Container Security Guide, vol. 800. Gaithersburg, MD, USA: NIST, 2017, p. 19

[11] S. Vaucher, R. Pires, P. Felber, M. Pasin, V. Schiavoni, and C. Fetzer, ''SGX-Aware container orchestration for heterogeneous clusters,'' in Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS), Jul. 2018, pp. 730–741.

[12] R. Buyya et al. (2017). ''A manifesto for future generation cloud computing: Research directions for the next decade.'' [Online]. Available: https://arxiv.org/abs/1711.0912.

[13] 451Research. (2017). 451 Research Says Application Container Market to Reach 2.7 Billion by 2020. Accessed: Oct. 22, 2017 [Online]. Available:https://www.enterpriseai.news/2017/01/10/451-research-saysapplication-container-market-reach-2-7-billion-2020/

[14] D. Walsh. (2014). Are Docker Containers Really Secure?. Accessed: Dec. 27, 2017. [Online]. Available: https://opensource.com/business/ 14/7/docker-security-selinux

[15] B.M.Abbott,''A security evaluation methodology for container images,'' M.S. thesis, Brigham Young Univ., Provo, UT, USA, 2017.

[16] T. Combe, A. Martin, and R. Di Pietro, ''To docker or not to docker: A security perspective,'' IEEE Cloud Comput., vol. 3, no. 5, pp. 54–62, Sep./Oct. 2016.

[17] A. Bettini. (2015). Vulnerability exploitation in Docker container environments. FlawCheck, Black Hat Europe, Netherlands. [Online]. Available: https://www.blackhat.com/docs/eu-15/materials/eu-15-BettiniVulnerability-Exploitation-In-Docker-Container-Environments.p

[18] Sari sultan et al.(2019). ''Container Security: Issues, Challenges and the road ahead.'' IEEE Access, 2019.