Analysis of Security in Docker Containers and Systems

Term Paper

Tvarita Jain[*], Srishti Mishra[†],
Mentor: Mr. H L Phalachandra
Department of Computer Science, PES University
Bangalore, India
Email: [*]tvarita1297@gmail.com, [*]srishtimishra56@gmail.com,,

Abstract— **The need for lightweight, efficient and secure Virtualization technology has increased with the growing popularity of cloud applications and cloud services. Docker, a new containerization tool provides an alternative to Virtual Machines with benefits of increased efficiency, reduced operating costs, faster build times, rapid scalability, portability and improved disaster management and recovery. Research into the security aspect of Dockers is critical for users to protect the confidentiality, integrity and availability of their applications from malicious actors and threats. Current research into Docker container security, from an internal and external perspective, have brought several vulnerabilities and weaknesses to light and pave the way for development of stronger defenses to ensure secure Docker environments. Docker currently supports the Linux hardening capabilities, resource-limiting techniques and the isolation of filesystems, networks, devices and processes as part of its Internal security. External security analysis of the Docker ecosystem assesses vulnerabilities in the Docker Hub - a cloud repository service for sharing Docker images - using a scalable Docker image vulnerability analysis (DIVA) framework and explores propagation of vulnerabilities through images. Docker systems in deployment scenarios face active threats by malicious attackers and research in this field culminated in a hopping defense mechanism, where the application instances are periodically moved from one host to another to ensure that attackers cannot keep control for extended periods of time. Studying the effect of internal and external security defenses on performance and QoS and continuously strengthening the Docker ecosystem is vital for Dockers to secure the cloud environment.**

I. Introduction

With the advent of cloud applications, the use of virtualization technology has increased dramatically, bringing benefits such as increased efficiency, reduced operating costs, rapid scalability, portability and improved disaster management and recovery. Container-based virtualization and hypervisor-based virtualization are the primary types of virtualization technologies currently in use. The security and performance of these systems remains a key concern for cloud providers and users alike, as cloud-based systems are connected to the Internet and are open to malicious cyber-attacks. In order to protect the confidentiality, integrity and availability of the system a variety of different defenses, including newer hardware with built-in security features, increasingly secure network protocols, and expensive malware detection software are employed. Today, it is vital that virtualization systems provide a secure environment to deploy cloud-based applications and protect their users from security threats.

II. Overview of Virtualization Systems

A. Hypervisor-based

Hypervisor-based virtualization : This type of virtualization provides solution at the hardware level, A hypervisor integrates a complete virtual layer on top of the Host operating system. These virtual machines comprise of the application, it's dependencies, guest OS and a separate kernel. Hypervisors are broadly classified as Type I and Type II hypervisor. Type I hypervisor or the bare metal hypervisor works directly on top of the underlying hardware of the host, for example, Xen. Type II hypervisor or the hosted hypervisor works on top of the host OS, Eg: KVM. [1]Performance comparison favors Type I over Type II. A prominent benefit of using Hypervisor-based solution for virtualization is it's ability to support a variety of environments which is not supported by container based solution. In a virtualization environment several virtual machines have security zones independent and inaccessible from other virtual machines. The hypervisor has the ability to secure the infrastructure of cloud. [2]Reasons for choosing this technology :- i) Hypervisor controls the hardware which is the only access path to it. This is the basis of it's secure infrastructure. ii) Since the hypervisor-based system is implemented below the guest OS in the Cloud Computing hierarchy, this shows that an attack passing through the security system if the guest OS is easy to detect. iii)It's used as a layer of abstraction that separates the virtual environment from the hardware underneath. iv)It simplifies the transaction-monitoring process in the cloud environment. [2] In conclusion, if a security manager can address Authentication, Authorization and Virtual Hardware and hypervisor security , cloud clients are well on the way to a comprehensive security policy.

B. Container-based

Docker is an open source platform that automates the deployment of applications inside lightweight software containers. Docker containers package the dependencies required by an application into a standard Docker Image, from which a Docker container is built. Containers provide a custom environment for your application but do not emulate the hardware layer nor install a complete operating system as virtual machines do, thereby eliminating the overheads of virtualization. Docker runs isolated environments on top of the operating system of the host machine using features of the Linux kernel such as cgroups to limit resources and namespaces for network, process and and filesystem isolation. However, the tight coupling of docker containers with the host operating system increases the attack surface for malicious actors. Applications running inside a Docker container run similarly to native processes on the host, albeit on a extra layer of abstraction but without the expensive emulation of the host operating system. As mentioned by Rad, Bhatti and Ahmadi [3], the main advantages of docker are its reduced startup time, portability, rapid delivery and scalability. Docker containers can easily coordinate with 3rd-party softwares, which aid in orchestrating and managing Docker containers. With the advent of cloud computing, popularity for Dockers has steadily increased due to the reduced overheads, easy configuration process, and quick deployment into cloud-based environments.

C. Performance Comparison: Virtual Machines vs Dockers

Dockers have made a prominent effect in the virtualization world. The key factors to this prominent growth are: speed, portability, scalability, rapid delivery , and density.

1) Speed
   Being light-weight compared to VMs , containers can be built much faster. This is because it does not incorporate an entire virtualized operating system. This results in faster development , testing , and deployment. Containers can be pushed for testing once they have been built and then from there , on to the production environment. It is also worth noting that a containerized application usually starts in a couple of seconds, on the other hand virtual machines take a longer time.
2) Portability
   Applications that are built inside the docker container can be easily moved as a single entity. The performance remains the same , or in extreme cases takes minimal hits. By default the applications running inside VMs cannot access hardware like graphics cards on the host, Containerized applications can. Multi-national companies like Nvidia have already evolved from this functionality.
3) Scalability
   When the preference boils down to scalable solutions, Container-based solution is favorable. Docker has the ability that it can be deployed in several physical servers,data servers, and cloud platforms. It can also be run on

every Linux machine. They can be easily transported from a cloud environment to local host and vice versa at a fast pace. It's also flexible to adjustments. Containers also use up only as many system resources as they need at a given time. VMs usually require some resources to be permanently allocated, hence they tie up resources on the host even if they aren't using them. Therefore, containers provide optimal resource distribution.

4) Density
Dockers use resources that are available more efficiently, more containers can be run on a single host. It's not preferable to run concurrent VMs on a single host as they require a lot of computation power and use up all of the resources that are already permanently allocated to them. Thus, dockers have more density and no overhead wastage of resources boosting their overall performance.

Dockers do provide a better performance solution relative to hypervisor-based systems , although it is necessary that the possibility of security issues should be evaluated.

III. Docker Security Analysis

While dockers provide a lightweight, easy to set up solution to automate deployment of applications in cloud-based environments, they come with a variety of pressing security concerns. Dockers consist of 4 main components: Docker Client and Server, Docker Images, Docker Registries and Docker Containers, which gives rise to two broad categories of security threats - External and Internal. The external aspects of Docker security consist of the Docker Client and Server, Docker Images (found on Docker Hub) and Docker Registries while the internal aspect is primarily concerned with the Docker Container and Docker Daemon.

Docker containers are part of a complex Docker ecosystem with Orchestrator and third-party instruments which automate deployments and testing, which increases the vulnerabilities exposed to attackers.

Security concerns in deployment scenarios where a single system or network of systems run multiple container deployments are higher and more at risk since the security of this system can be compromised by an attack on a single container.

A. *Internal security aspect*

Dockers run directly on the host operating system which exposes a larger attack surface to threat actors. Furthermore, Docker containers use Linux kernel features such as cgroups and namespaces, which require root privileges on the host when starting up the Docker daemon, exposing vulnerabilities and files of the host operating system.

The system and attacker model where a subset of containers are compromised, the attacker can perform various types of attacks, such as DOS and Privilege escalation.

To encounter with these attacks, Isolation and host hardening mechanisms are discussed.

1) Isolation -
   a) Process Isolation :- The idea is to prevent compromised containers from using process management interfaces to interfere with other containers. Dockers wrap these processes into namespaces and isolate these processes by limiting their permissions and visibility.
   Linux OSs generally contain a global process tree (which contains a reference to every process currently running on the system - parent-child hierarchy) We can now have nested process trees (use clone() system call to create a new PID namespace), where the child process tree will not know about the existence of parent/sibling process trees and hence is in its own process tree environment.

   b) Filesystem Isolation :- Filesystems must be protected from illegitimate access and modification. To prevent interference on other containers dockers use filesystem namespaces. However containers inherit the view of these filesystems from the host and this can be evaluated as a threat. Dockers can limit this in two ways, first by removing write permission to these filesystems from containers and second, by not allowing any process to remount any filesystem within the container.

c) Device Isolation :- Device whitelist controller of cgroups provides means to limit the set of devices accessible to dockers. Nodev ensures that pre-created device nodes do not communicate with the kernel. Docker grants access only if a container is 'Privileged'.

d) Network Isolation :- To prevent network based attacks like Man-in-the-Middle and ARP spoofing, Dockers use network namespaces. Thus, each container has its own IP addresses, routing tables , network devices, etc. The Ethernet bridge that implements the connectivity between containers and the host machine is vulnerable to flooding attacks since the bridge forwards all the incoming packets with no filtering.

e) Hardening Ports :- The docker container can be started with all it's ports closed by default until the user explicitly opens them.

2) Limiting of resources
:-In a Denial-of-Service attack, A process or a group of processes attempt to consume all the resources of the system. With the help of Cgroups in Linux, resource isolation can be implemented.This ensures that each container obtains its fair share of the resources.
3) Docker and Kernel Security System :-
To harden the security of a Linux host system , two popular mechanisms are in practice -

a) Linux Capabilities
Superusers are considered as the privileged users an OS, specifically speaking of Linux. The privileges of these superusers are divided into capabilities, which can be enabled or disabled by the kernel. Dockers use this feature to configure the capabilities of containers. By default, a large number of these capabilities are disabled to make sure that any vulnerability of acquiring the root privileges in a container will not affect the host kernel.

b) SELinux
Linux system provides Access Control mechanisms for objects in a system. SELinux enhances these mechanism by providing an access layer of mandatory access control. This mechanism controls everything using labels assigned to isolated objects. The access control rules are assigned to these labels as policies. The difference between this mechanism and the DAC mechanism is that , in SELinux the kernel is responsible for the enforcement of policies and not the object owners.This helps in prevention of the attacker acquiring control over the object if the owners are compromised.

B. External security aspect

An integral part of Docker systems is Docker Hub, a cloud registry service to share Docker Images. Docker images are useful for sharing specific application software along with supporting libraries and configuration files among users. Docker images are distributed using name-spaced repositories; Official repositories from vendors and Community repositories by any user or organization. The repository system allows versioning and forking, for example, new Docker images can inherit from each other in a parent-child relationship, adding versatility to Docker environments.

The complexity of software configuration in Docker Hub images, combined with a large number of images built by various parties, results in a significantly vulnerable landscape [4]. Analysis of the security vulnerabilities in the images by Rui Shu, Xiaohui Gu and William Enck [4] using the Docker Image Vulnerability Analysis (DIVA) System Framework uses the following approach

1) Image discovery: Use a name crawler to query Docker Hub and record repository names, both Community and Official, between the years 2008 to 2015. An existing web engine could have been used to identify repositories, along with randomized search strings to overcome the search results limit.
2) Image vulnerability analysis: Examines the downloaded images for vulnerabilities using Clair [5], which identifies insecure packages from the Common Vulnerabilities and

Exposures(CVE) vulnerability database [6], Ubuntu CVE Tracker [7], Red Hat Security Data [8] and other such databases and uses their qualitative and quantitative scores.

3) Inter-Image dependency analysis: Examines images for parent-child relationships via a directed graph, i.e an image dependency graph, and identifies the propagation of security vulnerabilities between images on Docker Hub.

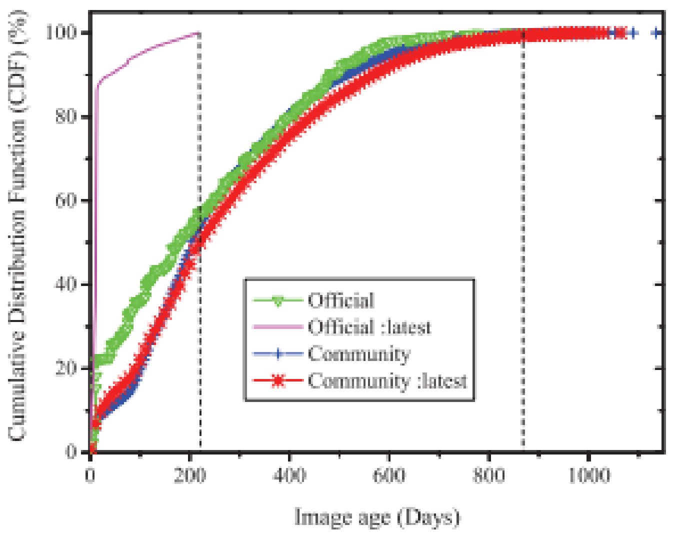


Fig. 1. Cumulative distribution function (CDF) of percentage distribution of the age of images at the time of analysis[4]

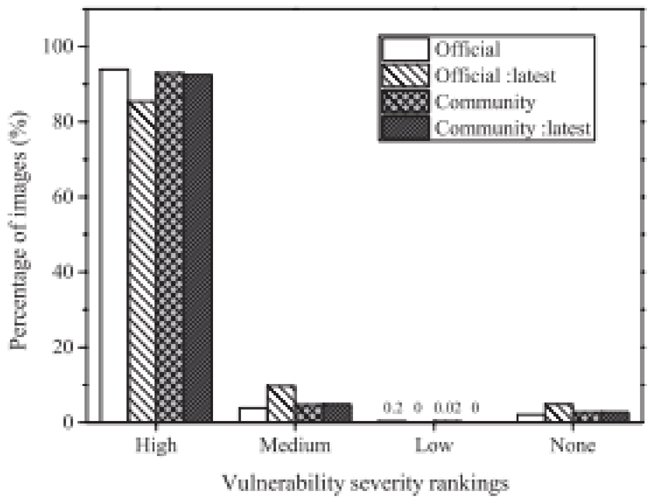
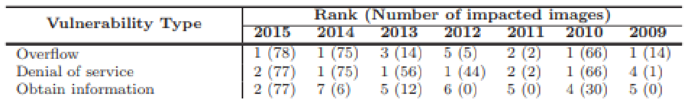


Fig. 2. Distribution of images based on most severe vulnerability[4]

The results from the DIVA system framework indicate that the average number of security vulnerabilities in the latest Official images were much lesser as compared to all Official images which indicate efforts towards better maintenance for official images, however, the number was about the same in both classes of Community images, possibly due to vulnerability propagation from older Community images. "A Study of Security Vulnerabilities on Docker Hub" [4] ranks vulnerabilities in terms of severity mapping the CVSS score from the [9] CVE details database to the NVD ranking, based on thresholds into 3 standard categories: "Low", "Medium", "High". As exhibited by Figure 2, most images - both Official and Community - contained at least a few vulnerabilities of high severity.

The study made the assumption that an image that has not been updated in a long time is more likely to contain more vulnerabilities, however this assumption fails to cover the case of stable secure versions which need not be repeatedly updated. As seen in Figure 1, it was found that overall 70% of images were updated within 400 days of the experiment, and nearly 86% of the Official images were updated recently. In terms of very recently updated images, only 10% of the Community images were updated very recently as compared to 20% of the Official images in results published by Rui Shu and Xiaohui Gu [4]. Images which have not been updated for a long period could be due to software compatibility issues, keeping snapshots of a runtime environment, or used to reproduce bugs in specific experimental environments.



Vulnerability Type	Rank (Number of impacted images)						
	2015	2014	2013	2012	2011	2010	2009
Overflow	1 (78)	1 (75)	3 (14)	5 (5)	2 (2)	1 (66)	1 (14)
Denial of service	2 (77)	1 (75)	1 (56)	1 (44)	2 (2)	1 (66)	4 (1)
Obtain information	2 (77)	7 (6)	5 (12)	6 (0)	5 (0)	4 (30)	5 (0)

Fig. 3. Vulnerability types ranked per year by the number of impacted :latest official images [4]

Analysis of the vulnerabilities found in Docker Hub indicate:

- Number of vulnerabilities: The number of CVEs found in the analyzed images grows steadily across the years, with 6,845 unique CVE vulnerabilities in the set of all community images and 1,554 unique CVE vulnerabilities in the set of all official images from the year 2008 to 2015.
- Types of vulnerabilities: CVEs are categorized into several types, according to the CVE Details database as shown in Figure 3. Figure 3 shows that 66 of the 93 official images contains an overflow vulnerability from 2010 in its latest version. Research by Rui Shu and

Xiahui Gu discovered that the vulnerabilities found 2010 onwards were caused by 2 unique CVEs; a significant vulnerability CVE-2010-4051 which was related to a RE-DUP-MAX overflow, and could lead to denial of service[4]. Furthermore, Community images were found to have a greater variety of vulnerabilities, however this could be due to the substantial amount of Community images analyzed as compared to Official images.

- Most Vulnerable packages: Specific packages, such as `glibc`, `util-linux`, `shadow`, etc were found to be the main offenders, affecting over 80% of the images in most categories[4] . This indicates that a small number of packages contribute to most of the vulnerabilities, and perhaps secure customized versions of these could be released to improve the security of Docker images.
- Propagation of vulnerabilities: On an average, child images inherit 80 or more vulnerabilities from their parents, regardless if the parent image is Official or Community, and frequently introduce new vulnerabilities themselves. [4] Rui Shu indicates that security updates are not applied on installing new software packages. The results from the image dependency graph correlated with top vulnerable packages exhibited a strong relationship with the top influential base images, demonstrating that the root cause of pervasive vulnerabilities on Docker Hub is the result of propagation from a relatively small set of highly influential base images.[4]

Dockers are a popular choice for deploying web application cloud clusters as lightweight and isolated containers. The security of the applications depends not only on the internal Docker security defenses but also on the deployment scenario. A cloud cluster of Docker containers can be used a security defense mechanism and increase security of the whole application via techniques such as hopping. The hopping defense mechanism periodically moves application instances from one host to another, changing the port and IP configurations etc, so that attackers cannot keep control of compromised parts of the application for long periods of time. However, the hopping technique impacts the Quality of Service (QoS), reliability of the system and might even crash the cluster, which was a prevalent problem when using Virtual Machines. Docker containers can easily be brought down and restarted on other hosts with faster startup times, portability, scalability and high density as discussed in the performance overview. Research by Weichen Wang analyzes the performance impact of hopping mechanism of Docker clusters experimentally on MongoDB clusters using the throughput of write requests as the key performance metric along with response times and other information. The work uses an automated process using load testing tools such as Apache JMeter and conducted on Amazon infrastructure with 64-bit Ubuntu 14.04 systems, 1024MB of RAM, and 2 CPU cores[10].

The baseline for the experiment was established on running a MongoDB cluster on the above setup with no containers being hopped.

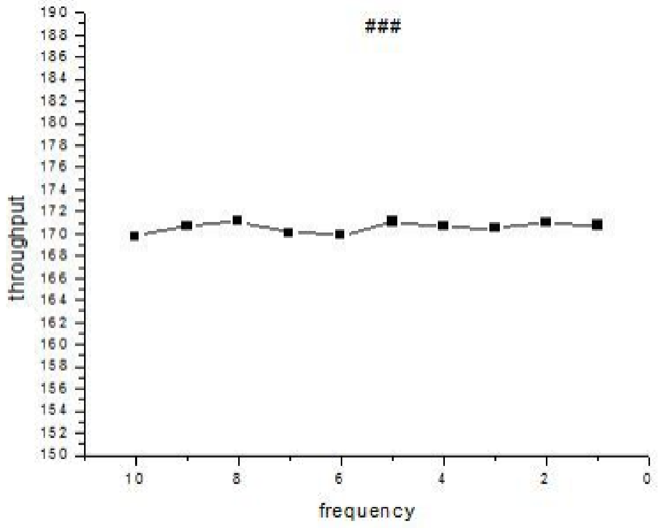


Fig. 4. Single container hopping at varying frequencies[10]

Performance impact against this baseline performance is examined regarding 3 key areas:

- Frequency of hops: As shown in Figure 4 by Weichen Wang, 1 to 5 containers were hopped at varying frequencies and resulted in achieving 50% of the baseline performance when up to 80% of the cluster nodes were hopped at 1-second intervals[10].
- Number of containers simultaneously being hopped: In the experiment, the frequency of the hops was fixed and the number of containers were changed. The results in ”A cyber-security defense method using Docker containers” [10]

exhibited a negative correlation of throughput with increase in hopping frequency and a reducing throughput as larger percentages of the cluster's containers were simultaneously hopped.

IV. Conclusion

Container-based virtualization can provide higher density virtual environments and better performance than hypervisor-based virtualization. However, the latter is argued to be more secure than the former , The results from vulnerability analysis of Docker Hub, an integral part of the Docker ecosystem, suggest that image publishers, users and maintainers alike need to focus on improving the Docker ecosystem by assessing vulnerabilities to avoid propagation and update security patches for images regularly by examining the latest security threats. Furthermore, development should focus on building customized Docker-secure versions of the most vulnerable packages or restrict their functioning through Docker internals. The work done by Rui Shu, Xiaohui Gu and William Enck [4] demonstrates a strong need for automated and systematic methods of applying security updates to Docker images.

The security level of Docker containers could also be increased if the operator runs them as"non-privileged" and enables additional hardening solutions in Linux kernel, such as SELinux. Dockers in deployment scenarios using hopping defense mechanism prove to be faster and provides considerable performance improvements as compared to Virtual Machines. The experimental results by Weichen Wang's work on Docker containers running a MongoDB cluster provides valuable insight into optimizing performance of Docker deployment systems. Further work in this field would benefit from experiments carried out using a variety of application softwares, on a considerable range of operating systems and with varying amounts of system resources in order to explore the factors on performance from a broader perspective.

References

[1] T. Bui, "Analysis of docker security," January 2015.

[2] F. Sabahi, "Secure virtualization for cloud environment using hypervisor-based technology," *International Journal of Machine Learning and Computing*, vol. 2, 2012.

[3] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An introduction to docker and analysis of its performance," *IJCSNS International Journal of Computer Science and Network Security*, vol. 3, 2017.

[4] R. Shu, X. Gu, and W. Enck, "A study of security vulnerabilities on docker hub," in *CODASPY '17 Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, no. 7, 2017.

[5] Coreos clair. [Online]. Available: https://github.com/coreos/clair

[6] Cve: Common vulnerabilities and exposures. [Online]. Available: https://cve.mitre.org/

[7] Ubuntu cve tracker. [Online]. Available: https://launchpad.net/ubuntu-cve-tracker

[8] Red hat security data. [Online]. Available: https://www.redhat.com/security/data/metrics/

[9] Cve details database. [Online]. Available: http://www.cvedetails.com/

[10] W. Wang, "A cyber-security defense method using docker containers," 2015.

[11] T. Combe, A. Martin, and R. D. Pietro, "To docker or not to docker: A security perspective," *IEEE Cloud Computing*, vol. 3, 2016.