Evaluation of Docker Container Networking

Muhammad Adnan Shah
201764@students.au.edu.pk
Cyber Security Department, Air University, Islamabad

***Abstract*—Docker is an open source platform with a client-server architecture providing a set of services for developing, running and shipping software applications. It has revolutionized the world of software development by providing platform as a service along with the virtualization provided through containers. Containers are lightweight portable package bundles containing all the required files and configurations for a software application. There can be many containers on a single docker host and all the communication is done through the networking protocols and drivers provided by docker. There are different network communication scenarios in docker, for example container-to-host, host-to-host, container-to-container with same host and container-to-container with different hosts. These network configurations come with their own security risks and concerns which need to be investigated and evaluated for possible threats and vulnerabilities.This paper gives a brief overview of docker networking configurations, container communication scenarios and their respective solutions provided by Docker. In particular it gives a details attack scenario and methodology to evaluate and exploit the Dynamic Host Configuration Protocol (DHCP) service of docker.**

***Index Terms*—Docker Security, Docker Engine, ARP Poisoning, MAC Spoofing, Docker Containers, Docker Hub**

I. Introduction

With the growth of virtualization technologies the need for a scalable and a secure user environment has also increased. Two main categories came forward with the solution in the market, one of them being virtual machines with hypervisor-based virtualization and the other one as containers based upon the container-based virtualization [1].

Virtual Machines: Virtual machines are based on the first type of virtualization technique and are very well known in the field. Provide virtualization on the hardware level and each virtual machine comes along with its own operating system, commonly known as "Guest OS". This increases the size and lowers the performance as well. Hypervisor based virtualization supports a variety of environments which container-based virtualization is unable to achieve i.e. one can not run windows containers in an environment based on Linux containers [2].

Containers: The container-based virtualization performed way better in terms of performance, flexibility and scalability. It has some edge to the hypervisor based virtualization due to its structure and architecture. However it was less secure as compare to hypervisor-based virtualization. Containers need less resources in order to setup and run the applications as they don't have an extra guest operating system like virtual machines. The world is shifting from hypervisor-based virtualization techniques to the container-based virtualization due to the difference in performance, scalability and ease of use. It's a fact that containers are faster and use lesser storage and resources as compared to virtual machines [3].

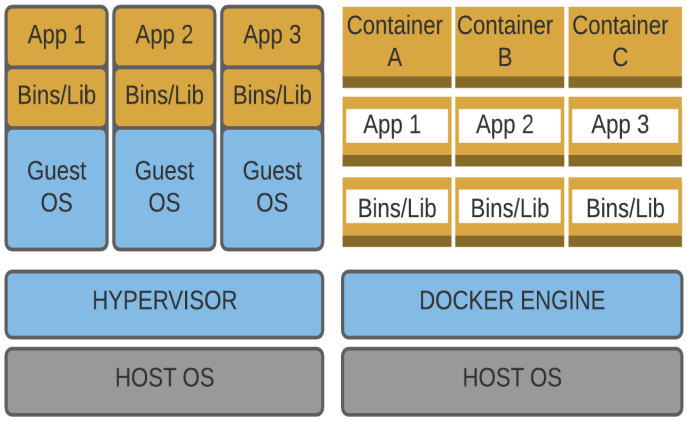

Fig. 1: Virtual Machines (Left) Vs Docker Containers (Right)

II. Docker Overview

Docker provides platform-as-a-service for deploying, running and shipping software applications. It uses container technology for achieving virtualization. Containers are lightweight virtual machines which help in isolating the environment. Docker has resolved a critical issue faced by developers and operation teams in the software development usually described as *It works on my machine*. This issue is resolved in docker as it packs all the required binaries in a single container which is portable and can be deployed on any machine. Docker provides a simple and secure way to create and manage containers very efficiently. It helps in the automation of processes after the deployment of applications. It provides a lightweight environment for the applications to run code smoothly, test the code and deploy the code to the production environment. The main components of docker are docker engine, docker hub and docker containers [4].

Docker Engine: Docker engine has a similar architecture as compare to the container-based virtualization architecture. In docker the management duty for all the containers is fulfilled by *Docker Daemon* and all containers run on top of it. On the other end *Docker Client* is responsible for providing user interface. Users run commands on it, which are sent to the docker daemon and use it for interacting with containers. Docker engine is one of the most important component of docker and it is considered to be a lightweight packaging tool which is both lightweight and portable. It can be run on the

different host as well as on the same host. Use of docker engine provides flexibility and scalability to docker due to its performance and applicability on different hosts [5].
Docker Hub: Docker hub is a central repository for storing container images, their testing and distribution on a cloud-based repository. The images provided on docker hub can be open source or public which are available to anyone on the platform and private repositories. we can call it a cloud based docker image or registry platform for accessing and posting container images [6].

III. Docker Networking

Docker provides its own networking solutions for establishing and maintaining communications between containers and hosts. The containers are isolated entities which neither have information about nor have any access to the other containers in the host. However they need to communicate with each other and with the outside world for sharing information and offer or receive services. Docker achieves this functionality with its *Network Drivers* [7].

A. Container-To-Container Same Host

A single docker host can have multiple containers. These containers may need to communicate with other for services or data exchange. **Bridge Network Driver** is the default network driver that is used to establish the communication between standalone containers inside a docker host.

B. Container-To-Host

Containers also need to communicate with the host, especially when they need some information from the outside world. For example if a container has a web application then all of its communication will go through host. **Host Network Driver** is usually used for this purpose. It helps to optimize the performance in case where containers use a wide range of ports and they don't require any network address translation *NAT*.

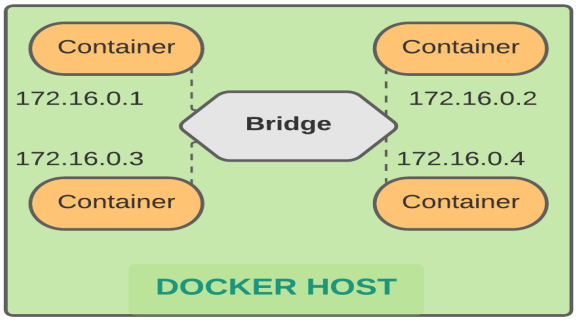

Fig. 2: Container-To-Container with Bridge Driver

C. Host-To-Host

Docker hosts also communicate with each for exchange of information and services. This is achieved by an **Overlay Network**. It is used to connect multiple docker hosts which gives them ability to communicate with each other.

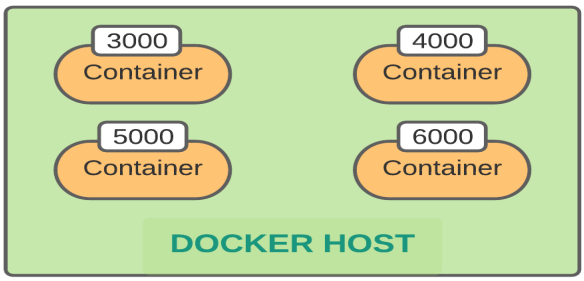

Fig. 3: Container-To-Host with HOST Driver using Ports

D. Container-To-Container on Different Host

Containers sometime need to communicate with the containers on another host to collaborate and run their services. For example web server container on one host and the database server container on another host need to communicate with each other to offer web app services, data storage and to perform information updates. These are also using the **Overlay Network**.

E. Other Drivers

There are some other drivers as well which provide the networking functionality for the containers and hosts. **Macvlan Driver** connects container interfaces directly with the host interfaces without using the port mapping or Linux bridge [8].

IV. Docker Network Vulnerabilities

Docker provides different networking facilities by use of different drivers like bridge, host, none, overlay, swarm and Macvlan. It uses Linux network namespaces, UTS and cgroups for achieve isolation of the network and provide efficient communication between containers and hosts. Namespaces like pid, mnt, ipc, UTS and net are used to restrict containers from the excessive use of shared resources like memory, kernel space and other resources [9].

A. Denial Of Service (DoS)

It is a very common attack which exhausts the network resources and result in partial or complete breakdown and halt of the system and services. In container based environment DoS resembles to the exhaustion of shared resources between containers. Docker is also structured in this way to share kernel space, memory and other resources between all the containers. So, in case of the excessive use of one resource by a container results in the unavailability of the resources to all other containers [10].

B. Attacks on Address Resolution Protocol (ARP)

Address resolution protocol (ARP) is responsible for MAC address translation to IP address for communication over the network. ARP table contains the pair wise entries for MAC address and IP address. This binding of physical and network address gives new identity to the machine or system trying to

connect with the other devices, especially over the internet. In the beginning ARP messages were not filtered as their is no built-in security mechanism inside the ARP protocol. In a docker environment, containers can imitate other containers which may lead to man in the middle attack (MiTM) and other security risks like ARP spoofing, ARP Denial of Service attacks and ARP cache poisoning attack [11].

V. Docker DHCP

Dynamic host configuration protocol (DHCP) is the most efficient protocol for providing connectivity to the internet. It assigns IP address to the machines and systems based upon their MAC address. DHCP contains its own network table which has the information about IP address allocation having pair wise entries for the IP and MAC address of each host/machine. It works with two routing tables including static table which has the static entries for IP address allocation and dynamic table which provides lease for IP address against the MAC for a limited time [12]. Let's look at how the DHCP works and the respective possible attacks.

A. DHCP Address Allocation

DHCP protocol offers IP addresses to hosts and keeps record of their identity over the network. There is a four step transaction involved in the process of providing IP addresses to the systems, which they can use as an identity to send and receive messages over the network. In the first step the client broadcasts a discover request to locate the DHCP server. The server responds with a broadcast message offering the services. Now the client has found or located the DHCP server and it sends a unicast request for the IP address from the server. The serve sends a unicast ACK message as an acknowledgement of the client request. At this point server sends the DHCP parameters to the client needed for configuring the client network [13].

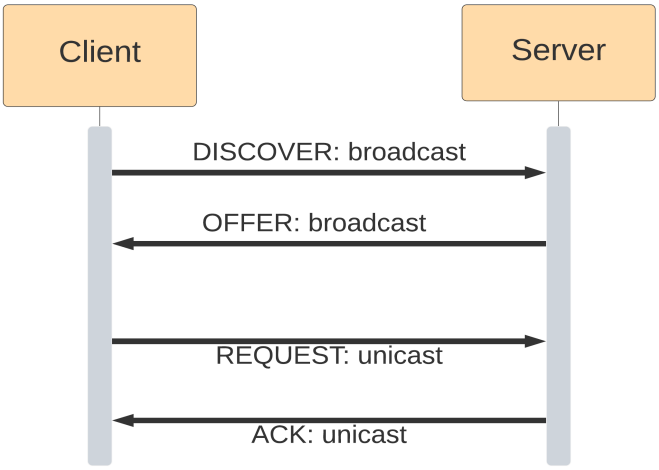

Fig. 4: DHCP Address Allocation Steps

B. DHCP Attack Scenarios

Docker contains network namespaces for isolation of resources like IP routing tables, other network devices and network protocols etc. The containers are allotted IP addresses and connected to the *bridge*. In this way there is not filtering mechanism or checks applied by the docker and it just forwards the traffic. This scenario gives rise to different vulnerabilities like MAC flood attack, ARP poisoning and man in the middle (MiTM) attack [14]. There are usually two main attack scenarios for attacking a DHCP server or service on a system or network. First is DHCP spoofing and second is the DHCP starvation attack.

1) DHCP Spoofing: In this scenario a simple client user which used by the attacker acts as a DHCP server and responds to DHCP requests. It records or maintains a list of IPs that are connected with the DHCP. This is same as the record maintained by DNS server or a default gateway that has a list of host IPs for sending the communication and packets through the network. In this way the all the traffic and communication is routed through the fake DHCP or hacker's computer. With this method the attackers are able to use their machines in order to attack the legitimate DHCP server and can also intercept and inspect the network communication [15].

2) DHCP Starvation: The main purpose of this attack is to exhaust the IP space of a DHCP server. The attacker sends numerous requests to DHCP for the IP address allocation and one by one takes all the possible IPs present in the DHCP server pool. This creates a situation similar to Denial of Service, specific to the lease of IP addresses. So, in this way the other systems can not get the IP addresses and hence DHCP services are unavailable to everyone.

VI. Conclusion

Docker provides virtualization of environment with the help of container-based technology. It also provides a rich environment for the development operation teams to develop and test the applications on a single platform. Docker helps to bind all the required binaries and libraries in a single bundle or docker image which is portable i.e. it can be run on any machine with docker client. In this paper we explored different network configurations provided by docker. We covered different scenarios of communication between docker containers and host operating systems. We evaluate these network configurations for possible security risks and threats. DHCP is one of the most important service over the network that implements IP and MAC address binding. It provides MAC to IP translation, IP to MAC translation, maintains the record, lease the IP addresses and renew the lease. However in docker the DHCP service is open to threats like DHCP spoofing and DHCP starvation. These attacks can also lead to DHCP snooping attacks, it exhibits a switch like behavior which does not filter any packet or communication and forwards all the traffic.

REFERENCES

[1] A. M. Potdar, D. Narayan, S. Kengond, and M. M. Mulla, "Performance evaluation of docker container and virtual machine," *Procedia Computer Science*, vol. 171, pp. 1419–1428, 2020.

[2] "Can i run windows server container on linux host operating system." [Online]. Available: https://forums.docker.com/t/can-i-run-windows-server-container-on-linux-host-operating-system/6713

[3] T. Bui, "Analysis of docker security," *arXiv preprint arXiv:1501.02967*, 2015.

[4] A. SHARMA and R. HUSSAIN, "A study, analysis and deep dive on docker security vulnerabilities and their performance issue."

[5] Docker, "Docker engine," 2021. [Online]. Available: https://docs.docker.com/engine/

[6] TechTarget, "Docker hub," 2017. [Online]. Available: https://searchitoperations.techtarget.com/definition/Docker-Hub

[7] S. S. Hegde and P. Jayarekha, "Basic analysis of docker networking." *International Journal of Advanced Research in Computer Science*, 2017.

[8] Dockerlabs, "Introduction to macvlan," 2019. [Online]. Available: https://dockerlabs.collabnix.com/intermediate/macvlan.html

[9] H. Zeng, B. Wang, W. Deng, and W. Zhang, "Measurement and evaluation for docker container networking," in *2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE, 2017, pp. 105–108.

[10] A. Tomar, D. Jeena, P. Mishra, and R. Bisht, "Docker security: A threat model, attack taxonomy and real-time attack scenario of dos," in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2020, pp. 150–155.

[11] R. Yasrab, "Mitigating docker security issues," *arXiv preprint arXiv:1804.05039*, 2018.

[12] N. K. Jotani, "Dynamic host configuration protocol service," *International Journal of Wireless Network Security*, vol. 3, no. 1, pp. 28–30, 2017.

[13] WikiPedia, "Dynamic host configuration protocol," 2021. [Online]. Available: https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol

[14] J. Wenhao and L. Zheng, "Vulnerability analysis and security research of docker container," in *2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE)*. IEEE, 2020, pp. 354–357.

[15] T. C. L. Network, "Describe dhcp spoofing attacks," 2020. [Online]. Available: https://learningnetwork.cisco.com/s/question/0D53i00000Ksumt/describe-dhcp-spoofing-attacks