Docker Security: A Threat Model, Attack Taxonomy and Real-Time Attack Scenario of DoS

Aparna Tomar
Department of Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
Email: aparnatomar29@gmail.com

Diksha Jeena
Department of Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
Email: dikshajeena02@gmail.com

Preeti Mishra
Department of Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
Email:dr.preetimishranit@gmail.com

Rahul Bisht
Department of Computer Science and Engineering
Graphic Era Deemed to be University
Dehradun, India
Email: me.rahul.bisht@gmail.com

***Abstract*—As the last decade experienced an explosion in the development and use of virtualization technologies, the need for an efficient and secure virtualization solution has also been increased. All the solutions that emerged can be classified into two major classes i.e. hypervisor-based virtualization and container-based virtualization. Container technologies have been around for a very long time but Docker is a relatively new and the most dominant candidate among all the other technologies. Along with so many advantages, it has a few disadvantages as well in which its security is the primary and the most crucial concern. In this paper, we propose a threat model for Docker with all the possible attack scenarios in Docker-based host systems. Furthermore, the paper also provides a detailed classification of attacks that can take place on various layers of Docker along with the description of each one of them. Lastly, the paper presents a real-time case study on Denial of Service (DoS) attack in the Docker environment.**

***Index Terms*—Containers, Virtual Machines (VMs), Docker, Security, Attacks**

I. Introduction

The last decade saw a dramatic increase in the development and use of virtualization technologies. Hence, the demand for an efficient and secure virtualization solution (that can provide scalable, portable, dense and secure user environment) has also increased. A large number of such solutions emerged which can be classified into two major classes i.e. hypervisor-based virtualization and container-based virtualization (also known as OS-level virtualization) [1] [2]. The latter one emerged as a lightweight alternative to VMs and it provides more lightweight and efficient user environments. Moreover, containers take less start-up time hence they are better in performance than VMs. In the year 2016, the value of the container market was 762 million dollars and it is expected to reach 2.7 billion dollars in the year 2020 [3]. There are many container technologies available like Linux Containers (LXC), Linux-Vserver, Open VZ, with Docker being the relatively new and the most dominant candidate.

Docker is definitely the future of virtualization technologies. Developers and users will surely replace traditional virtualization technologies with Docker as they would know more about it and its potentiality. Docker has many advantages like speed, scalability, portability, density and rapid delivery [4]. However, there exist some disadvantages as well. The disadvantages of Docker includes no complete virtualization (as it depends on the Linux kernel which is provided by the local host) and not being able to run on older machines (supports only 64-bit machines) [4]. When the disadvantages of Docker comes into the picture, its security is the primary and the most crucial concern. In the VMs, applications can only communicate with the VM kernel and there exists no communication with the host kernel. So in order to attack the host kernel, the attacker first needs to bypass the VM kernel and the hypervisor. But in all the container technologies, like Docker, it is not the case. In Docker, applications can directly access and communicate with the host kernel thus permitting the attacker to hit the host kernel directly [1]. This is one of the key reasons that raise security concerns about Docker.

With security being such a major issue, there are many attacks possible in the Docker environment on its various components like hosts, applications, containers, and Docker engine. Some of the possible attacks are Denial of Service (DoS), ARP spoofing, Man-in-the-Middle (MitM) attack, poisoned images and so on. In this paper, we address these attacks by providing a threat model.

As per the best of our knowledge, there is no work done which covers the layered threat model, detailed attack taxonomy and real-time case study of an attack scenario.

In this paper, we propose a layered threat model that contains all the possible attack scenarios in a Docker-based host system. We also provide a detailed classification of attacks that can take place on various layers of Docker (like container, application, Docker engine and host). Lastly, we provide a case study of Denial of Service (DoS) attack in Docker-based environment.

The major contributions of this paper can be summarized as below:

- To propose a layered-threat model with attacking scenarios in Docker-based virtualization environment.
- To propose a detailed classification of attacks in Docker-based virtualization environment.
- To propose a real-time case study of DoS attack in a multi-core Docker based host system.

The rest of the paper is organized as follows: Section 2 gives the details of the related work being done in the field of docker. Section 3 describes the layered threat model being proposed with all the possible attack scenarios. Section 4 gives the classification of attacks and their description. Section 5 provides the case study of DoS attack in Docker environment. Finally, section 6 concludes the paper.

II. Related Work

There exist some other papers such as Bui [1], Combe et al. [9], Yasrab [2], Sultan et al. [3] where Docker security has been addressed. The work by Rad et al. [4] provided a brief description about the four main components of Docker i.e. Docker client and server, Docker images, Docker registries, and Docker containers, along with the advantages and disadvantages of Docker. They further presented a study on Docker performance in comparison to other virtualization technologies like Kernel-based Virtual Machine (KVM), Linux Containers(LXC) and XEN. This study is based on the studies being done in previous papers (Seo et al. [5], Scheepers [6], Felter et al. [7]).

Seo et al. [5] found out that Docker performance is better than KVM. Felter et al. [7] proves that there is no major difference between Docker and KVM when it comes to wastage of resources. Moreover, KVM was found out to be faster than Docker. Scheepers [6] found out that LXC takes a longer time to perform tasks in comparison to Xen Server. LXC is better as it wastes fewer resources whereas Xen is better in terms of equally distributing resources.

On the other hand, the work by Martin et al. [8] presented the overview of Docker ecosystem along with the brief description of its main components i.e. specification, kernel support, Docker daemon, Docker hub, Docker store and dedicated OS. Typical Docker use-cases under which recommended, wide-spread and cloud providers CaaS use-case are described. Moving forward, an adversary model for three use-cases is defined along with a vulnerability-oriented analysis.

Bui [1] described container-based virtualization and hypervisor-based virtualization, which are the two main types of virtualization technologies along with Docker overview under which Docker engine, Docker container and Docker hub are described. Furthermore, the security level of Docker is analyzed based on two areas: (1) the internal security of Docker, and (2) how Docker interacts with the security features of the Linux kernel.

Combe et al. [9] presented an overview of the Docker ecosystem and docker security. Further, they provided docker usages challenges under which the Docker usages has been distinguished into three types i.e. recommended usages, wide-spread usages, and PaaS providers usages. Lastly, some vulnerabilities affecting Dockers usages are described.

Yasrab [2] presented a brief comparison between VMs and Docker along with the Docker overview and its security analysis. Furthermore, some attacks (like DoS, container breakouts, poisoned images, MitM, and ARP spoofing) and how to protect Docker from these attacks is also provided. Lastly, under the proposed solution for a safe Docker environment, access control policy modules and deployment guidelines are described.

A recent work by Sultan et al. [3] provided a comparison between virtual machines and containers. Furthermore, a threat model is given for the containers with four use cases which cover the security requirements of the host-container threat landscape. Among these four use cases, the first three relies on software-based solutions, whereas the last one relies on hardware-based solutions. Under the software-based solutions, Linux kernel features (like namespaces, control groups, secure computation mode abbreviated as seccomp) and Linux Security Modules (LSMs) are discussed. In hardware-based solutions, description of virtual Trusted Platform Modules (vTPM) and Intel SGX is being provided. At last, some open issues and future research directions are given.

III. Proposed Threat Model

Several attack scenarios can be derived for docker security so a detailed list of all such scenarios is provided to give a better understanding to the reader.
In the attack scenarios, the words trusted, semi-trusted and untrusted are used for different Docker components. A *semi-trusted* adversary is a passive adversary that can help in the gathering of information but will not deviate from any protocol execution. Whereas an *untrusted* adversary is an active adversary that might deviate from protocol execution in order to gather information [3].

The proposed threat model with all the possible use cases are shown in Fig. 1 and they are described below:

Attack Scenario I: Application To Container Attack
Here we assume that if control policies are set then applications cannot break them and root privileges might be required by some applications. If an application gains control over the control manager, the host system and other containers within the system can get compromised, which make it a very important use case [3].
Hence, we should protect containers from the applications running inside them.

Attack Scenario II: Container To Container Attack
Here the assumption is, one or more containers along with the applications inside them are semi-trusted or untrusted. The goal here is to protect a container from all the other

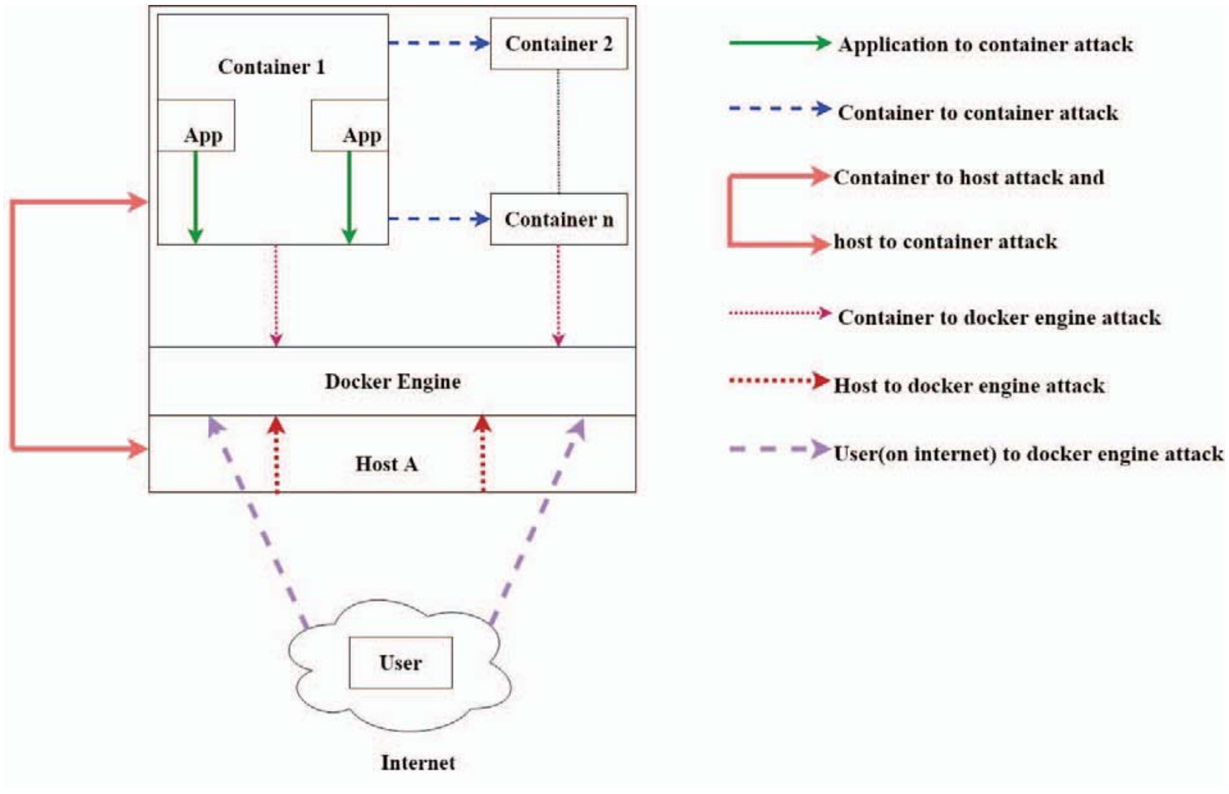

Container 1
App
App
Container 2
Container n
Docker Engine
Host A
User
Internet
Application to container attack
Container to container attack
Container to host attack and
host to container attack
Container to docker engine attack
Host to docker engine attack
User(on internet) to docker engine attack

Fig. 1: The Proposed Layered Threat Model

containers. If we make the containers behave like VMs (i.e. they do not know anything about each other, unless required) then it would be a perfect case for this attack scenario [3].

Attack Scenario III: Container To Host Attack
Here we assume that at least one container within the host is semi-trusted or untrusted. We also assume that the applications which are inside as well as the applications in honest containers can also be semi-trusted or untrusted. The goal here is to eliminate all such scenarios where a container can target the confidentiality, integrity, and availability of host components. If we make containers act like VMs then that would be a perfect scenario [3].

Attack Scenario IV: Host To Container Attack
Here the assumption is, the containers are trusted but the host can either be semi-trusted or untrusted. With the emergence of Container as a Service (CaaS), running containers on untrusted hosts should be avoided [3]. Several attacks including both active and passive attacks can be launched from semi-trusted and untrusted hosts against the containers running inside them. Active attacks are more harmful in comparison to passive attacks as they are capable enough to change the behavior of the applications.

Attack Scenario V: Container To Docker Engine Attack
Container can easily attack docker engine and to prevent the docker engine from this, access rights plays a vital role i.e. access rights (in order to control the containers) should be given only to the trusted users.

Attack Scenario VI: Host To Docker Engine Attack
As the host is having all the privileges, so it can easily attack the docker engine. All such scenarios need to be eliminated where the host can misuse its privileges.

Attack Scenario VII: User (On Internet) To Docker Engine Attack
If the hosts are present in a cloud environment then docker engine can be attacked by using the Cloud Container Attack Tool (CCAT) [10]. We can run containers on AWS by using the Amazon Elastic Container Service (ECS).

IV. Attack Taxonomy

In this section, attacks are classified based on attack layers i.e. the layer where the attack can take place. The chosen attack layers are as follows: 1) Container 2) Application 3) Docker engine 4) Host. The description of the attacks is given below and the layers where these attacks will take place are shown in Fig. 2.

1. Malware
In this attack, the attacker tricks the victim in order to open a webpage which is controlled by the attacker. The attacker can

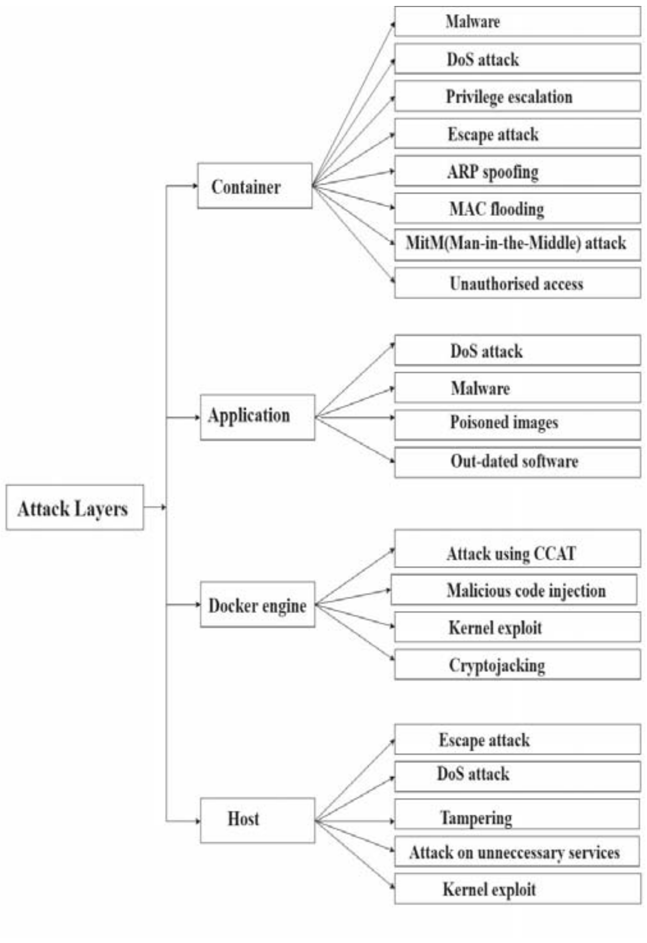


Fig. 2: Attack Taxonomy

then execute the Docker Build command to create a container that will execute an inconsistent code and this will be done by using a REST API [11]. By using Host Rebinding technique, the attacker will be able to bypass Same-Origin Policy protections and can gain access to the underlying host machine. The attacker can then get the malware running on the developers machine or he/she can inject it into the Docker image thus allowing it to spread every time the Docker is launched.

2. Denial of Service (DoS) Attack

There are two types of DoS attack, i.e. resource exhaustion and disability. In resource exhaustion, the attacker overly consumes the resources like CPU and memory. Whereas in disability, the attacker can cause kernel panic or application crashing by sending a large number of requests [12]. In containers based architecture, all containers share kernel resources. DoS can take place when one container exploits a particular resource, thus making the other containers starve for it [2].

3. Privilege Escalation

In this attack, the attacker gains the kernel root privilege. Two methods are used to achieve this attack, i.e. memory modification and file modification. In memory modification, the attacker overwrites certain data structure in the memory, thus resulting in the change of control flow. In file modification, the privileged files will be modified by the attacker in order to change the password of the superuser or modify file attributes to execute malicious programs with root privileges [12].

4. Container Escape Attack

In order to access necessary resources, an attacker must have root privileges inside the container. If this attack gets successful, the attacker will gain full control of the host and can try to attack the resources present in the network. This can allow the attacker to even deploy a malicious container inside the environment [13].

5. ARP spoofing and MAC Flooding Attacks

Virtual Ethernet bridge is used to provide connectivity between the container and the host machine. Using this approach, Docker creates a virtual ethernet bridge which automatically forwards the packets between all its network interfaces. Whenever a new container is created by Docker, a new virtual Ethernet interface (with a new name) is established and get connected with the bridge. The default connectivity model of Docker is vulnerable to ARP spoofing and MAC flooding because all the incoming packets are forwarded by the bridge to the desired interfaces without performing any filtering [1].

When an attacker sends fake ARP messages over a local area network, it is called ARP spoofing. In this attack, the attacker will link his/her MAC address with the IP address of an authorized user and he/she will start getting each and every piece of information from that IP address. In such a situation, the attacker will be able to sniff secret information shared between web application and containers and will also be able to inject a malicious payload into the network [2].

6. Man-in-the-Middle (MitM) Attack

During such attacks, a malicious user inserts himself/herself in the communication between two non-malicious parties and tries to monitor, alter and steal valuable information getting shared between the two parties [2].

7. Unauthorized access

A malicious user getting access to a legitimate Docker system can cause a lot more damage than we can even think of. Right from attacking all the containers on the Docker to attacking the host system itself, unauthorized access can cause harm in so many ways. Hence, any kind of unauthorized access should be prevented.

8. Poisoned Images

Injecting virus or trojan infected software or running outdated and vulnerable versions of software can cause the problem of poisoned images. In Docker, downloaded image is verified based on the presence of a signed manifest, but the downloaded image checksum from the manifest never gets authenticated from Docker. Because of this, an attacker with a signed manifest can transmit any image which can lead to serious vulnerabilities [2].

9. Out-dated Software

Out-dated software has many issues which get fixed in the updated versions. So, the running of out-dated software can cause serious security issues. Thus, out-dated software should be avoided.

10. Attack Using CCAT

Cloud Container Attack Tool (CCAT) was created to test the security of cloud environments. An attacker can use this tool to abuse the AWS credentials for further exploitation in the AWS environment [10]. We can run containers on AWS by using the Amazon Elastic Container Service (ECS). Amazon ECS is a highly scalable, high performance container management service.

11. Malicious Code Injection

One of the great security issues in containers is that an attacker could infect a container by injecting a malicious program into it, that could escape and attack the host system. A new vulnerability was found [14], which allow a malicious container to overwrite host runc (open source command line tool for running containers) library and thus gain root level privileges on the host. To do this, the attacker needs to place a malicious container within our system which is not a very difficult task to perform.

12. Kernel Exploit

The applications running inside containers can get compromised and exploit in case of a kernel-level exploit. As all the containers share the same kernel, so if any application gets hijacked and obtain some privileged rights of the kernel, then all the running containers as well as the host platform can get compromised [2].

13. Cryptojacking

In this attack, the malware variants will rob the CPUs of infected machines to steal computational power in order to mine for virtual coins such as Ethereum (ETH) and Monero (XMR) [15]. These cryptocurrencies are later sent to wallets controlled by attackers.

14. Tampering

Insecure container configurations lead to the risk of tampering for the host systems. If a container is allowed to mount sensitive directories on the host OS, the container can later change the files in those directories. These changes can cause an impact on the security and stability of the host and all the other containers running on it [16].

15. Attack on unnecessary services

Services are programs that hear and reply to network traffic. Some services also allow direct access to our computer such as FTP servers, web servers, file servers, email and proxy servers, open ports and so on. Unneeded and unsecured services can lead to an open door for attackers [17]. On the other hand, the more services running on our computer, the more opportunities will be there for others to use them, break into them and take control of the computer through them.

V. CASE STUDY

System Configuration

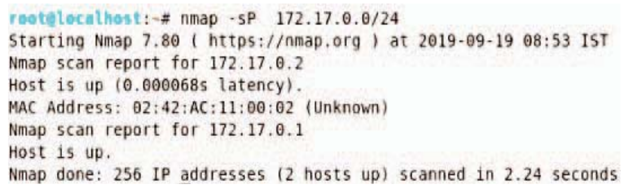
The system performing the attack has Kali Linux 2019.2 installed in it. The system is having main memory of 8 GB and hard disk storage of 1 TB out of which 400 GB is allotted to Kali Linux and the rest is allotted to Windows. The CPU of the system belongs to the family of 5th generation Intel Core i5. The Docker installed in the system is of community edition having the version 19.03.02.

Attack Scenario

Attacker and victim are present on the same network and attack is being done from Kali Linux to container inside which some web server images were running.*hping3* is used to perform the attack.

Steps involved

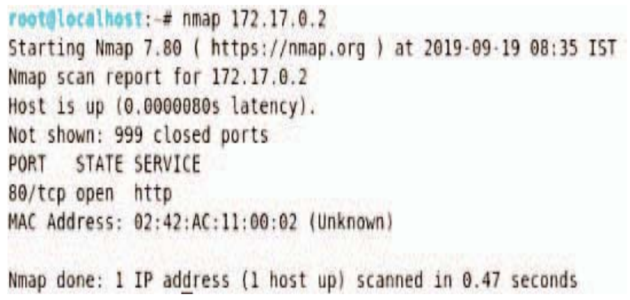
1.Scan the system for available hosts, using nmap as shown below in Fig. 3.



```
root@localhost:~# nmap -sP  172.17.0.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-19 08:53 IST
Nmap scan report for 172.17.0.2
Host is up (0.000068s latency).
MAC Address: 02:42:AC:11:00:02 (Unknown)
Nmap scan report for 172.17.0.1
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 2.24 seconds
```

Fig. 3: Scanning the system for available hosts

2. After knowing about the available hosts, well check for open ports on those hosts as shown in Fig. 4.

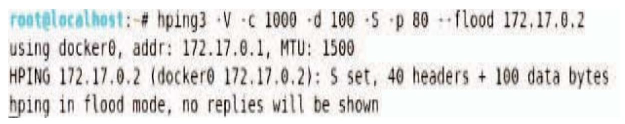


```
root@localhost:~# nmap 172.17.0.2
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-19 08:35 IST
Nmap scan report for 172.17.0.2
Host is up (0.0000080s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 02:42:AC:11:00:02 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds
```

Fig. 4: Scanning the hosts for open ports

3. Lastly, we will use the hping3 tool to perform DoS using the following command as shown in Fig. 5.
hping3 -V -c 1000 -d 100 -S -p 80 –flood 172.17.0.2

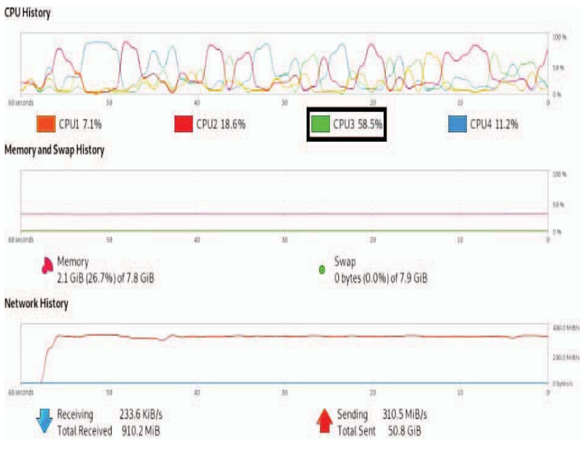


```
root@localhost:~# hping3 -V -c 1000 -d 100 -S -p 80 --flood 172.17.0.2
using docker0, addr: 172.17.0.1, MTU: 1500
HPING 172.17.0.2 (docker0 172.17.0.2): S set, 40 headers + 100 data bytes
hping in flood mode, no replies will be shown
```

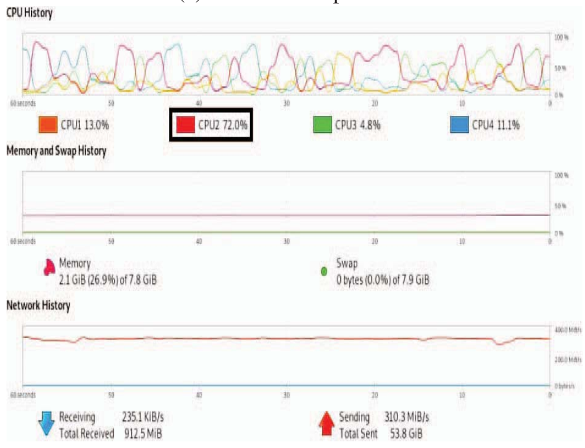
Fig. 5: Performing DoS attack

If we interpret this command into words, it means that 1000 packets (-c 1000) at a size of 100 bytes (-d 100) each are being sent. The SYN Flag (-S) should be enabled. To perform the attack on victims HTTP web server, port 80(-p 80) has been specified and the -flood flag has been used to send packets as fast as possible. The IP address provided at the last is the victim's IP address.

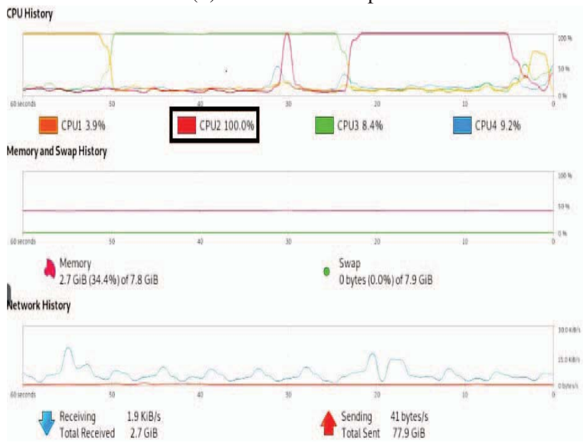
After the attack has been done, there will be a sudden increase in CPU' usage. This is being shown in Fig. 6.



(a) On the third processor



(b) On the second processor



(c) On the second processor

Fig. 6: Impact of DoS attack on various processors

VI. Conclusion

Virtualization technologies have been around since a very long time but Docker is a relatively new and the most dominant candidate. As developers and users will know more about its potentiality, it will definitely become the future of virtualization technologies. It has many advantages like speed, portability, density and rapid delivery. Despite having these many advantages, its security is still a major concern. In this paper, we proposed a threat model for all the possible attack scenarios that can take place in a Docker-based host system. We further provided a detailed classication of attacks that can take place on various layers of Docker (like container, application, Docker engine and host). The DoS attack has also been performed in the Docker environment (using nmap and hping3) and the case study is provided for the same. In future, we will expand our threat model for more than one host and will also propose an efcient technique to deal with some of the Docker-based attacks.

References

[1] T. Bui, "Analysis of Docker Security," CoRR, vol. abs/1501.02967, 2015. [Online]. Available: http://arxiv.org/abs/1501.02967
[2] R. Yasrab and I. Technology, "Mitigating Docker Security Issues."
[3] S. Sultan, I. Ahmad, T. Dimitriou, "Container Security: Issues Challenges and the Road Ahead", IEEE Access, vol. 7, pp. 52976-52996, 2019
[4] B. B. Rad, H. J. Bhatti, M. Ahmadi, "An Introduction to Docker and Analysis of its Performance", IJCSNS International Journal of Computer Science and Network Security, vol. 17, no. 3, March 2017.
[5] K.-T. Seo, H.-S. Hwang, I.-Y. Moon, O.-Y. Kwon, B.-J. Kim, "Performance comparison analysis of linux container and virtual machine for building cloud", Advanced Science and Technology Letters, vol. 66, no. 105111, pp. 2, 2014.
[6] M. J. Scheepers, "Virtualization and Containerization of Application Infrastructure: A Comparison," 2014.
[7] W. Felter, A. Ferreira, R. Rajamony, J. Rubio, "An updated performance comparison of virtual machines and linux containers", Performance Analysis of Systems and Software (ISPASS) 2015 IEEE International Symposium on, pp. 171-172, 2015.
[8] A. Martin, S. Raponi, T. Combe, R. Di Pietro, "Docker ecosystemvulnerability analysis", Computer Communications, vol. 122, pp. 30-43, 2018.
[9] T. Combe, A. Martin, 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
[10] "Exploiting AWS ECR and ECS with the Cloud Container Attack Tool (CCAT),"Rhino Security Labs, 27-Aug-2019.[Online]. Available: https://rhinosecuritylabs.com/aws/cloud-container-attack-tool/. [Accessed: 15-Sep-2019].
[11] S. Nichols, "Malware? In my Docker container? It's more common than you think," The Register - Biting the hand that feeds IT, 28-Jul-2017.[Online].Available: https://www.theregister.co.uk/2017/07/28/malware_docker_containers/. [Accessed: 15-Sep-2019]
[12] X. Lin, L. Lei, Y. Wang, J. Jing, K. Sun, Q. Zhou, "A measurement study on linux container security: Attacks and countermeasures", Proc. 34th Annu. Comput. Secur. Appl. Conf., pp. 418-429, Dec. 2018.
[13] "CVE-2019-5736: RunC Container Escape Vulnerability Provides Root Access to the Target Machine," CVE-2019-5736: RunC Container Escape Vulnerability Provides Root Access to the Target Machine - Security News - Trend Micro BE. [Online]. Available: https://www.trendmicro.com/vinfo/be/security/news/vulnerabilities-and-exploits/cve-2019-5736-runc-container-escape-vulnerability-provides-root-access-to-the-target-machine. [Accessed: 16-Sep-2019].
[14] S. J. Vaughan-Nichols, "Doomsday Docker security hole uncovered,“ ZDNet, 12-Feb-2019. [Online]. Available: https://www.zdnet.com/article/doomsday-docker-security-hole-uncovered/. [Accessed: 15-Sep-2019].
[15] C. Osborne, "This is how Docker containers can be exploited to mine for cryptocurrency,“ ZDNet, 21-Jan-2019. [Online]. Available: https://www.zdnet.com/article/this-is-how-docker-can-be-exploited-to-covertly-mine-for-cryptocurrency/. [Accessed: 15-Sep-2019].
[16] M. Souppaya, J. Morello, K. Scarfone, Application Container Security Guide, Gaithersburg, MD, USA:NIST, vol. 800, pp. 190, 2017.
[17] Turn Off Unnecessary Services. [Online]. Available: https://its.ucsc.edu/security/services.html. [Accessed: 15-Sep-2019].