Suresh Gyan Vihar University Journal of Engineering & Technology
(An International Bi-Annual Journal)
Volume 8, Issue 1, 2022 , pp. 28-42
ISSN: 2395-0196

A novel Framework for Docker and Container Security and their risk Assessment

Aakriti Sharma[1], Bright Keswani[2], Pankaj Kumar Gupta[3]
[1]Research Scholar, Suresh Gyan Vihar University, Jaipur
[2]Professor, Suresh Gyan Vihar University, Jaipur
[3]Associate Professor, Tirupati College of Technical Education, Jaipur

Abstract-The goal of our study is to automate the scanning of Docker images and test them for vulnerabilities, as well as summarise the findings of the scans in order to speed up the process of assessing the security of different systems that use Docker. What we're looking for is an automated tool that can scan Docker Hub's image collection for vulnerabilities and then provide a report on them. A predetermined list of vulnerabilities will be scanned to find any vulnerabilities in the library's most popular pictures. After that, we'll provide scan statistics to provide a clear picture of the vulnerabilities included in the images downloaded from Docker's repository. An initial prototype will be built and tested against images on the Docker Hub. The results of these tests will be reported for each image in an easy-to-read style to make the process of analysing a Docker image's security easier.

Introduction

When evaluating the security of a Docker container, it's important to focus on three main areas: the kernel's native defences, namespaces, and c-groups. Users can customise the container configuration profile in Docker, and the daemon presents an aggressively named surface, to attack [6]. As a result, the daemon is vulnerable to attack through different inputs, such as image loading from a hard drive (using the command 'docker load), CD, or Flash disc (using 'docker pull'). Kernel "hardening or coagulation" security and their interaction with containers enhances security. The Docker daemon will run Linux facilities, virtual networks, file systems, jobs, logs, subprocesses, and so on under the direction of the user. Within the containers, the Docker machine engine is put to use. SSH server and current monitoring/supervision procedures may be used to run Docker on a dedicated server and then move all extra services into the containers prescribed by Docker (NRPE, collected, etc). The Docker container platform has fast become the industry standard. As a result of Docker Hub's abundance of readily available Docker engine images, it is a popular choice among developers. There are, however, a number of Docker Hub images that include security flaws and liabilities. Organizations seek to analyse pre-fabricated containers for susceptibility, liability, propensity, and malware before allowing a third-party user to utilise them,

in order to learn more about the recipient. Similarly, enterprises want to place containers produced in-house through the same stringent protective software development lifecycle tests that web programmes go through before being deployed. FlawCheck is here. To ensure that no CI/CD processes using container registry in a product using [CI] continuous integration of services are vulnerable to ransomware or malware, this security automation tool was developed in the Private Registry for Docker. It checks all Docker container images and their workloads for these issues and more in all CI/CD processes by checking for vulnerabilities. For coders and action panels, it also offers a single plane of tumbler or console in order to create buoyancy before deployment. Providing IT while maintaining the appropriate level of security controls necessitates extensive tool capability and trust ability testing through pen testing. SSH, cron, syslogd, hardware admin tools (coded script-programmed automated load modules), wired/wireless network pattern tools (e.g., to regard DHCP, VPNs, and or WPA), etc. are all part of a server's root privileges.

Major security challenge of Docker are:

a) Linux NamespacesTo make it appear to processes within the namespace that they have their own isolated instance of a global system resource, a namespace wraps it in an abstraction Other processes in the namespace can see changes to the global resource, but all other processes cannot. Namespaces can be used to create containers. 2 Containers must have their own IP address, port, and routing to isolate the network in a distributed context. The /proc/net directory and other net resources may be found under the Net namespace. Semaphores, message queues, and shared memory are used for inter-process communication in containers. Communication between processes within a container is the same as communication between processes in the same IPC Namespace. Mount Namespace [8] provides quarantined file systems by quarantining file system mount points, and the system users can view all mounted file system in the current namespace through the command "cat /proc/[pid]/mounts", [9] and also the statistics of file device in mount namespace, including the mount file name, file system type, mount location, etc. A unique internal hostname and system version number are required for containers to identify themselves as distinct from the rest of the system. User namespace and PID namespace separate permissions and PIDs, ensuring the independence of users and processes within the container. The Linux Namespaces system is used by Docker to accomplish UTS, IPC, PID, Network, Mount, and User isolation. A process's namespace identification number may be retrieved using the command "sudo ls -l /proc/[PID]/ns" on Linux systems.

b) **Copy-on-Write**On the copy-on-write mechanism used by the Docker file system, a basic file system image can be shared by all containers. Once writing data to this file system, a copy of that data is made and then written to the

associated file. This mechanism not only significantly reduces resource consumption, but also effectively ensures data isolation among containers. The Power of 3.3 Traditional UNIX [10] implementations distinguish between privileged and unprivileged processes while executing permission checks. The effective user ID of a privileged process is 0, which is referred to as "superuser" or "root" (whose effective UID is nonzero). Unprivileged processes are subject to complete permission checks depending on the process' credentials, but privileged processes skip all kernel permission checks (usually: effective UID, effective GID, and supplementary group list). Using this approach, container root privileges may be segregated from those outside the container, making it easier to manage rights within the container.

c) **Cgroups :**Control Cgroups, usually referred to as Cgroups, are a Linux kernel feature which allow processes to be organized into hierarchical groups whose usage of various types of resources can then be limited and monitored. The kernel's Cgroups interface is provided through a pseudo-filesystem called CgroupFS. Grouping is implemented in the core Cgroups kernel code, while resource tracking and limits are implemented in a set of per-resource-type subsystems (memory, CPU, and so on). It is based on kernel hooks to realize resource scheduling, restriction and checking, and as the cornerstone of building a series of virtualization tools. Docker uses this mechanism to reduce the attack surface by restricting access by containerized applications to the physical devices on a host. Thus containers have no default device access and have to be explicitly granted device access, and can effectively defend against DoS attacks by consuming resources[11].

d) **Seccomp :**Secure computing mode is a Linux kernel feature that can be used to restrict the system call of processes in container. Limiting the kernel system call through default Seccomp strategy of Docker can reduce the risk of executing malicious code. Current default filtering system calls include open by handle at, bpf, mount, keyctl, ptrace and more, can effectively prevent against CVE-2014-3519, CVE-2015-8660, CVE-2016-0728 exploit attacks. For some vulnerable system calls out of the default Seccomp, there might be some risks of special exploit[12].

e) Image Security : Using Docker images, which may be found in public or private repositories, containers can be created. Docker users may access official images from the Docker community and ecosystem via the Docker hub. Officials have recommended a signature technique in the most recent version of Docker Datacenter to assure image security. Administrators can utilise The Update Framework (TUF) and Docker Material Trust (DCT) to set

up signature policies that prohibit untrusted content from being accessed and used on their systems. Because container images are constantly changing and need to be pushed and pulled across the infrastructure, securing communications is especially crucial while building and shipping applications. TLS is used for all communications with the registries to protect both the integrity of the content and the privacy of the information. An internal corporate CA root certificate can be added to the truststore[13] through the usage of the public PKI infrastructure.

Docker has developed as a lightweight substitute for virtual machines (VMs) that provides greater support for microservices architecture. $2.7 billion is estimated to be the market value of the docker market in 2020, compared to $762 million in 2016. Market studies suggest that container security is a major worry and obstacle to adoption for many firms, despite their widespread use in cloud computing developing domains such as service meshes. Study of docker security and solution literature. In order to speed up the process of assessing the security of various Docker-based systems, we're working to develop an automated technique for screening and testing Docker images for vulnerabilities [14]. What we're looking for is an automated tool that can scan Docker Hub's image collection for vulnerabilities and then provide a report on them. A predetermined list of vulnerabilities will be scanned to find any vulnerabilities in the library's most popular pictures. After that, we'll provide scan statistics to provide a clear picture of the vulnerabilities included in the images downloaded from Docker's repository. Our method will be put to the test by building a prototype and testing it against the images available on Docker Hub [15]. The results of the testing will be presented for each image in a way that makes it simple to understand how secure a Docker image is. [15]

2. Review of the work

We used articles from prestigious academic conferences, journals, and books for our project. There were times when we relied on research that was either unpublished or released in a non-commercial format, such as reports or policy statements. Because the container business dominates the research and publishes about it in several web venues, these papers have been included. Academic papers on Linux containers are few and few between, with little focus on container images or how to develop them. White papers and blog postings were the primary sources of information for this study. A healthy dose of common sense was exercised on this material before it was accepted for inclusion. Containers, according to many in the security community, need to be closely scrutinised because of the possibility of misuse. This study's findings have yet to be confirmed, but there is plenty of cause for alarm. As the first of its sort in academia, this study should not be overlooked. Containers are shown to be susceptible time and time again in this report. When it comes to protecting yourself and your loved ones, it's better to think ahead of time. The attackers have the upper hand if security is only addressed when issues arise.

Security features, solutions, threats, vulnerabilities, exploits, tools, standards,

assessment methodology, applications, and container alternatives were all considered in our decision process. Researching container security depended heavily on Google Scholar and terms like "Containers Security" or "Docker Security" were the most commonly used. After that, we deleted sources that were general to containers and those we had not discussed..

Various container management approaches have emerged since Docker's 2013 debut [16]. In contrast to other container technologies, Docker does not include any security measures for known security flaws in its images. An application-building system built on Docker will rapidly collapse since the images may be shared without a security fault detection process. A Docker Engine Vulnerability Detection System is being implemented in this post (DIVDS). The suggested DIVDS analyses a Docker image when its images are retrieved from the repository of Docker images.

R. Sairam and colleagues (2019) [17] The Internet of Things (IoT) is advancing because of the proliferation of intelligent devices and things. As a result, our lives have become increasingly dependent on these technological elements. Small-scale protective features and weaknesses in these systems are of great concern to these intelligent devices because of cyber thieves' advantage in complexity. Conventional centralised IT security techniques have limited scalability and expense. This type of smart gadget would function better if it could be managed at the edge of an IoT network, close to where it is located. On the network edge, various security measures can be applied to safeguard mobile devices in a smart home or corporate setting. Introducing network edge protection features necessitates the use of NFV, which we discuss in detail in this paper. To accomplish this, NETRA is developing a new lightweight, network-based docking architecture for IoT security virtualization features. Using the suggested design, we show how it has advantages in terms of memory utilisation and latency, as well as performance, average load and scalability, over the present NFV system. We evaluate the proposed NFV-based IoT protection edge detection and show that threats with more than 95% precision may be estimated in less than one second..

S. Sultan and colleagues (2019) [18] In order to better support the design of microservices, containers were created as a lightweight replacement for VMs. Containers' market worth is expected to grow from 762 million dollars in 2016 to 2.7 billion dollars in 2020. Container health is a major worry for many businesses and a barrier to adoption, despite their status as a simplified approach for providing micro applications in rapidly developing sectors like cloud storage and application meshes, according to business research. Our focus in this study is on container safety and solutions. The threat posed by host containers prompted us to create four widely used programmes to solve the security concerns. For example, one may use it to defend a container from programmes running inside it, or to protect the container itself from those apps, or to guard against containers attacking the host computer (IV). Our software-based solutions in the first three situations all rely on Linux kernel capabilities (e.g. nameplaces, CGroups, and seccomp), as well as protection modules (e.g. AppArmor). TPMs and other hardware-driven security solutions, such as trusted

device support, are the primary emphasis of the latter framework (i.e., Intel SGX). We expect that this evaluation will aid researchers in gaining a better knowledge of container security vulnerabilities and threats. In addition, we point out unanswered scientific questions and potential routes of research that might lead to greater research in this field.

According to J et al (2020) [19], C Diekmann, J and others For complicated systems like Docker, Linux containers are becoming more and more frequent. However, for distributed microservice deployment, the primitive safety of network access control is frequently overlooked or left to the network operations team. The application-specific security needs cannot be enforced satisfactorily by low-level network access control lists. No matter how well Docker and network operators operate together, they don't provide fine-grained networking access control between containers or application development. In a made-up narrative, we're keeping tabs on DevOp Engineer Alice as she works on a web application. We demonstrate the task that Alice is supposed to do and provide the necessary tool assistance from the beginning of the design and software engineering process through the network operations and automation stages.. As a DevOps complete stack, Alice is involved in high-quality design and network issues. By focusing on network access control and building out a tool-based solution, we have exposed the flaws in today's policy management. Academic research shows that a full stack engineer does not link many existing instruments between the various abstraction layers. With Isabell / HOL, our tools are open source and subject to regular evaluation.

Proposed Methodology

We drew on papers published in prestigious academic journals and publications as well as conference proceedings. Unpublished or published in non-commercial formats like reports, policy declarations, and the like were sources on occasion. Because the container business dominates the research and publishes about it in several internet sources, we've included articles on the topic of containers in our database. Academic papers on Linux containers are few and few between, with little focus on container images or how to develop them. Most of the information for this study came from white papers or blog posts obtained on the Internet. Before evaluating it for inclusion in this publication, the information was put through a rigorous process of common sense. Security experts generally agree that containers should be closely monitored because of the potential for misuse. While the bulk of the concerns raised in this study have yet to be found in reality, there are still plenty of reasons to be concerned about them all. In academia, this study is the first of its sort and should not be discounted. Throughout this report, proof of containers' vulnerability is shown. Preventative measures are better than reactive ones in the event of a problem, so keep that in mind. If you only talk about security after you have a problem, the attackers have already gained the upper hand.

Container security features, solutions, threats and vulnerabilities, as well as tools and standards for evaluating container alternatives were all considered in our decision process. To conduct our research, we mostly utilised Google Scholar and searched for "container security," "Docker

security," and "Linux containers," among other terms. Removed sources that were general to containers and were not discussed further.

Since the debut of Docker in 2013, container management projects have appeared in a variety of sectors. In contrast to other container technologies, Docker does not include any security safeguards for known security flaws in its images. As Docker images are interchangeable, Docker-based application-building systems are soon collapsed due to the lack of a security defect diagnostic in the exchange process. Vulnerability Diagnostics for Docker Engine are discussed in this post (DIVDS). Using the Docker picture repository and the planned DIVDS diagnostic tool, we can diagnose Docker images.

In a study by R. Sairam and colleagues (2019) [17] The Internet of Things (IoT) is in full swing, thanks to the proliferation of smart gadgets. This has a profound effect on our surroundings and makes our life reliant on these technological aspects. Intelligent objects are concerned about small-scale protective features and weaknesses in these systems, which help cyber thieves. There are limits to the scalability and cost effectiveness of traditional centralised IT security systems. As a result, intelligent devices like these are best managed at the edge of IoT networks, close to where they are located. Security features that can be applied in a smart home / business environment on the network edge are discussed below. In order to implement network edge protection features, we emphasise the significance of Network Virtualization Functions (NFV). Network-based docking architecture for IoT security virtualization is being implemented by NETRA to achieve this goal. According to our findings in terms of scalability and scalability and the present NFV design we show that the suggested architecture is superior in terms of capacity, memory utilisation, latency, performance and average load compared to current NFV architecture. We evaluate the proposed NFV-based IoT protection edge detection and show that threats with more than 95% precision may be estimated in less than one second.

In the work of S. Sultan and colleagues (2019) [18] Containers were created to be a more lightweight alternative to virtual machines (VMs) and to better support the microservices architecture. Containers' market worth is expected to grow from 762 million dollars in 2016 to 2.7 billion dollars in 2020.. Container health is a major worry for many firms and an acceptability barrier for micro applications, even if they are seen as a simplified manner of delivering them and play an important part in developing fields like as cloud storage and application meshes. In this paper, we explore the available research on container security and safety solutions. In response to the threat posed by host containers, we've created four widely used apps. A few examples of why containers could be useful include: (1) defending the container from applications within, (2) safeguarding the container within, (3) protecting the host from containers, and (IV). Our software-based solutions in the first three situations (e.g. nameplaces, CGroups, and seccomp) and protection modules rely heavily on Linux kernel capabilities (e.g. AppArmor). TPMs and other hardware-driven security mechanisms lie at the heart of the latter approach (i.e., Intel SGX). A deeper knowledge of potential vulnerabilities and assaults on container protection requirements is something we hope our

examination may provide. This isn't the only field of study that may benefit from further research; we also point out outstanding questions in science.

According to J et al(2020)[19], C Diekmann, J and others For complicated applications like Docker, Linux containers are becoming more and more commonplace. It's not uncommon for distributed microservice deployments to neglect or defer to the network operations team basic safety measures like network access control. Access control lists at the low levels of the network do not sufficiently enforce the security standards that are relevant to each application. As a result, neither Docker nor network operators enable the deployment of fine-grained network access controls between containers. While reading a fictitious novel, we're keeping tabs on DevOp Engineer Alice. We demonstrate the task that Alice is intended to perform and provide the necessary tool assistance, from the original design and software engineering to network operations and automation. As a DevOps complete stack, Alice is involved in high-quality design decisions and network problems. Policy management has been weakened by focusing on network access control and putting up a tool-based solution to this problem. According to our research, a full stack engineer does not link many existing instruments between abstractions. As part of Isabell / HOL, we've submitted our tools for peer review and made them open source. We drew on papers published in prestigious academic journals and publications as well as conference proceedings. Unpublished or published in non-commercial formats like reports, policy declarations, and the like were sources on occasion. Because the container business dominates the research and publishes about it in several internet sources, we've included articles on the topic of containers in our database. Academic papers on Linux containers are few and few between, with little focus on container images or how to develop them. Most of the information for this study came from white papers or blog posts obtained on the Internet. Before evaluating it for inclusion in this publication, the information was put through a rigorous process of common sense. Security experts generally agree that containers should be closely monitored because of the potential for misuse. While the bulk of the concerns raised in this study have yet to be found in reality, there are still plenty of reasons to be concerned about them all. In academia, this study is the first of its sort and should not be discounted. Throughout this report, proof of containers' vulnerability is shown. Preventative measures are better than reactive ones in the event of a problem, so keep that in mind. If you only talk about security after you have a problem, the attackers have already gained the upper hand.

Container security features, solutions, threats and vulnerabilities, as well as tools and standards for evaluating container alternatives were all considered in our decision process. To conduct our research, we mostly utilised Google Scholar and searched for "container security," "Docker security," and "Linux containers," among other terms. Removed sources that were general to containers and were not discussed further.

Since the debut of Docker in 2013, container management projects have appeared in a variety of sectors. In contrast to other container technologies, Docker does not include any security safeguards for known security flaws in its images. As Docker images are interchangeable,

Docker-based application-building systems are soon collapsed due to the lack of a security defect diagnostic in the exchange process. Vulnerability Diagnostics for Docker Engine are discussed in this post (DIVDS). Using the Docker picture repository and the planned DIVDS diagnostic tool, we can diagnose Docker images.

In a study by R. Sairam and colleagues (2019) [17] The Internet of Things (IoT) is in full swing, thanks to the proliferation of smart gadgets. This has a profound effect on our surroundings and makes our life reliant on these technological aspects. Intelligent objects are concerned about small-scale protective features and weaknesses in these systems, which help cyber thieves. There are limits to the scalability and cost effectiveness of traditional centralised IT security systems. As a result, intelligent devices like these are best managed at the edge of IoT networks, close to where they are located. Security features that can be applied in a smart home / business environment on the network edge are discussed below. In order to implement network edge protection features, we emphasise the significance of Network Virtualization Functions (NFV). Network-based docking architecture for IoT security virtualization is being implemented by NETRA to achieve this goal. According to our findings in terms of scalability and scalability and the present NFV design we show that the suggested architecture is superior in terms of capacity, memory utilisation, latency, performance and average load compared to current NFV architecture. We evaluate the proposed NFV-based IoT protection edge detection and show that threats with more than 95% precision may be estimated in less than one second.

In the work of S. Sultan and colleagues (2019) [18] Containers were created to be a more lightweight alternative to virtual machines (VMs) and to better support the microservices architecture. Containers' market worth is expected to grow from 762 million dollars in 2016 to 2.7 billion dollars in 2020.. Container health is a major worry for many firms and an acceptability barrier for micro applications, even if they are seen as a simplified manner of delivering them and play an important part in developing fields like as cloud storage and application meshes. In this paper, we explore the available research on container security and safety solutions. In response to the threat posed by host containers, we've created four widely used apps. A few examples of why containers could be useful include: (1) defending the container from applications within, (2) safeguarding the container within, (3) protecting the host from containers, and (IV). Our software-based solutions in the first three situations (e.g. nameplaces, CGroups, and seccomp) and protection modules rely heavily on Linux kernel capabilities (e.g. AppArmor). TPMs and other hardware-driven security mechanisms lie at the heart of the latter approach (i.e., Intel SGX). A deeper knowledge of potential vulnerabilities and assaults on container protection requirements is something we hope our examination may provide. This isn't the only field of study that may benefit from further research; we also point out outstanding questions in science.

According to J et al(2020)[19], C Diekmann, J and others For complicated applications like Docker, Linux containers are becoming more and more commonplace. It's not uncommon for distributed microservice deployments to neglect or defer to the network operations

team basic safety measures like network access control. Access control lists at the low levels of the network do not sufficiently enforce the security standards that are relevant to each application. As a result, neither Docker nor network operators enable the deployment of fine-grained network access controls between containers. While reading a fictitious novel, we're keeping tabs on DevOp Engineer Alice. We demonstrate the task that Alice is intended to perform and provide the necessary tool assistance, from the original design and software engineering to network operations and automation. As a DevOps complete stack, Alice is involved in high-quality design decisions and network problems. Policy management has been weakened by focusing on network access control and putting up a tool-based solution to this problem. According to our research, a full stack engineer does not link many existing instruments between abstractions. As part of Isabell / HOL, we've submitted our tools for peer review and made them open source.

Step for Implement the Proposed Work

1. Our Framework provide separate containers for each micro-service keeping Container image size small.
2. We used to put ssh inside container, “docker exec” can be used to ssh to Container.
    a. Use only signed Container images.
    b. Mount devices and volumes as read-only.
3. Run application as non-root. If root access is needed, run as root only for limited operations using features like Capabilities, Seccomp, SELinux/AppArmor.
4. Keep OS secure with regular updates. Using Container optimized OS is an option here since they provide automatic pushed update.
5. Store root keys, passphrase in a safe place and not expose in Dockerfile. Docker has plans to manage keys with Docker datacenter. Highest quality and security is maintained for the official images.

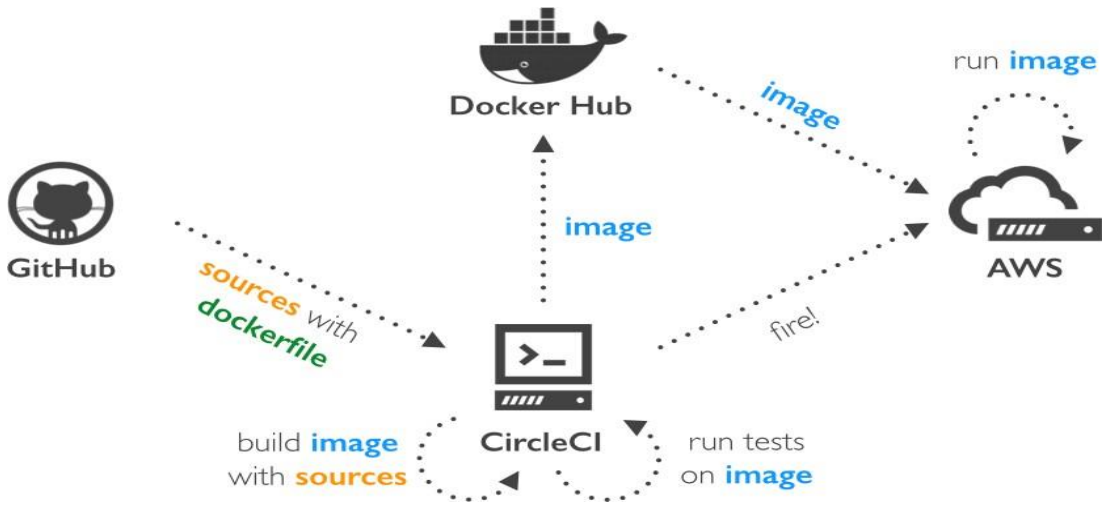

Figure 1: Proposed Framework for Docker security

Research Gaps identified in the proposed field of investigation

Companies have long deployed applications on virtual machines[36,37,38] (VMs) or bare metal servers. Security for that infrastructure involved securing your application and the host it's running on and then protecting the application as it runs. Containerization introduces several new challenges that must be addressed.

1.Containers enable micro services, which increases data traffic and network and access control complexity [39].

2.Containers rely on a base image, and knowing whether the image comes from a secure or insecure source can be challenging. Images can also contain vulnerabilities that can spread to all containers that use the vulnerable image[40,41].

3.Containers have short life spans, so monitoring them, especially during runtime, can be extremely difficult. Another security risk arises from a lack of visibility into an ever-changing container environment[42].

4.Containers, unlike VMs, aren't necessarily isolated from one another. A single compromised container can lead to other containers being compromised.

5.Containerized environments have many more components than traditional VMs, including the Kubernetes orchestrator that poses its own set of security challenges[43,44]. Can you tell which deployments or clusters are affected by a high-severity vulnerability? Are any exposed to the Internet? What's the blast radius if a given vulnerability is exploited? Is the container running in production or a dev/test environment?

6.Container configuration is yet another area that poses security risks. Are containers running with heightened privileges when they shouldn't? Are images launching unnecessary services that increase the attack surface? Are secrets stored in images? [45,46,47]

7.As one of the biggest security drivers, compliance can be a particular challenge given the fast-moving nature of container environments [48,49] . Many of the traditional components that helped demonstrate compliance, such as firewall rules, take a very different form in a Docker environment.

8.Finally, existing server workload security solutions are ill-equipped to address container security challenges and risks.

Conclusion

Deep Learning has been used extensively to identify variations in network attacks. Nonetheless, there have been no cases of network stability that have not been tested by implementations of various profound computing algorithms in real-time networks. Moreover, devices that can accommodate large-scale traffic need to be considered because of the introduction of high-performance computing. Given the rapid growth of network attacks, our Intrusion Detection Program (AI-IDS) was introduced and deployed. In order to adequately extract features of real time HTTP traffic without encryption, enropy compression calculation, We propose an Optimal Convolutionary Neural and Long Speed (CNN-LSTM) models, a normalized UTF-8 character encoding for Spatial Feature Learning. Repeated experiments in two public data sets (CSIC-2010, CICIDS2017) and fixed real-time data have demonstrated its excellent performance. AI-IDS distinguishes sophisticated attacks such as unknown patterns, coded, or fudged attacks from benign roads by training payloads that

analyze the true or false positives with a tool for labeling. This is a modular and scalable framework, built on Docker files, which distinguishes user-defined functions by separate pictures. It also contributes to writing and enhancing the Snort IDS rules based on newly identified patterns. The template could accurately analyze unknown web attacks by measuring the probability of maliciousness through continuous training.

References

[1.] S. Kwon and J. Lee, "DIVDS: Docker Image Vulnerability Diagnostic System," in *IEEE Access*, vol. 8, pp. 42666-42673, 2020. doi: 10.1109/ACCESS.2020.2976874

[2.] Q. Liu, W. Zheng, M. Zhang, Y. Wang and K. Yu, "Docker-Based Automatic Deployment for Nuclear Fusion Experimental Data Archive Cluster," in *IEEE Transactions on Plasma Science*, vol. 46, no. 5, pp. 1281-1284, May 2018. doi: 10.1109/TPS.2018.2795030

[3.] J. Bhimani *et al.*, "Docker Container Scheduler for I/O Intensive Applications Running on NVMe SSDs," in *IEEE Transactions on Multi-Scale Computing Systems*, vol. 4, no. 3, pp. 313-326, 1 July-Sept. 2018. doi: 10.1109/TMSCS.2018.2801281

[4.] P. Smet, B. Dhoedt and P. Simoens, "Docker Layer Placement for On-Demand Provisioning of Services on Edge Clouds," in *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 1161-1174, Sept. 2018. doi: 10.1109/TNSM.2018.2844187

[5.] S. Liu, Z. Xu and Z. Tian, "Implementation of NRF in the Docker-based NFV platform," in *The Journal of Engineering*, vol. 2019, no. 23, pp. 8884-8887, 12 2019. doi: 10.1049/joe.2018.9135

[6.] H. Jin *et al.*, "Architecture Modelling and Task Scheduling of an Integrated Parallel CNC System in Docker Containers Based on Colored Petri Nets," in *IEEE Access*, vol. 7, pp. 47535-47549, 2019. doi: 10.1109/ACCESS.2019.2909774

[7.] X. Guan, X. Wan, B. Choi, S. Song and J. Zhu, "Application Oriented Dynamic Resource Allocation for Data Centers Using Docker Containers," in *IEEE Communications Letters*, vol. 21, no. 3, pp. 504-507, March 2017. doi: 10.1109/LCOMM.2016.2644658

[8.] L. Ma, S. Yi, N. Carter and Q. Li, "Efficient Live Migration of Edge Services Leveraging Container Layered Storage," in *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 2020-2033, 1 Sept. 2019. doi: 10.1109/TMC.2018.2871842

[9.] Z. Lu, J. Xu, Y. Wu, T. Wang and T. Huang, "An Empirical Case Study on the Temporary File Smell in Dockerfiles," in *IEEE Access*, vol. 7, pp. 63650-63659, 2019. doi: 10.1109/ACCESS.2019.2905424

[10.] R. R. Yadav, E. T. G. Sousa and G. R. A. Callou, "Performance Comparison Between Virtual Machines and Docker Containers," in *IEEE Latin America Transactions*, vol. 16, no. 8, pp. 2282-2288, Aug. 2018. doi: 10.1109/TLA.2018.8528247

[11.] H. Sami, A. Mourad and W. El-Hajj, "Vehicular-OBUs-As-On-Demand-Fogs: Resource and Context Aware Deployment of Containerized Micro-Services," in *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 778-

790, April 2020. doi: 10.1109/TNET.2020.2973800

[12.] T. Mekonnen *et al*., "Energy Consumption Analysis of Edge Orchestrated Virtualized Wireless Multimedia Sensor Networks," in *IEEE Access*, vol. 6, pp. 5090-5100, 2018. doi: 10.1109/ACCESS.2017.2783447

[13.] C. Diekmann, J. Naab, A. Korsten and G. Carle, "Agile Network Access Control in the Container Age," in *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 41-55, March 2019. doi: 10.1109/TNSM.2018.2889009

[14.] P. D. Turnbull, T. W. G. Docker and C. J. Worsley, "Management education for effective technology transfer," in *Engineering Management Journal*, vol. 2, no. 5, pp. 237-245, Oct. 1992. doi: 10.1049/em:19920059

[15.] R. Sairam, S. S. Bhunia, V. Thangavelu and M. Gurusamy, "NETRA: Enhancing IoT Security Using NFV-Based Edge Traffic Analysis," in *IEEE Sensors Journal*, vol. 19, no. 12, pp. 4660-4671, 15 June15, 2019.doi: 10.1109/JSEN.2019.2900097

[16.] S. Kwon and J. Lee, "DIVDS: Docker Image Vulnerability Diagnostic System," in IEEE Access, vol. 8, pp. 42666-42673, 2020. doi: 10.1109/ACCESS.2020.2976874

[17.] R. Sairam, S. S. Bhunia, V. Thangavelu and M. Gurusamy, "NETRA: Enhancing IoT Security Using NFV-Based Edge Traffic Analysis," in IEEE Sensors Journal, vol. 19, no. 12, pp. 4660-4671, 15 June15, 2019. doi: 10.1109/JSEN.2019.2900097

[18.] S. Sultan, I. Ahmad and T. Dimitriou, "Container Security: Issues, Challenges, and the Road Ahead," in IEEE Access, vol. 7, pp. 52976-52996, 2019. doi: 10.1109/ACCESS.2019.2911732

[19.] C. Diekmann, J. Naab, A. Korsten and G. Carle, "Agile Network Access Control in the Container Age," in IEEE Transactions on Network and Service Management, vol. 16, no. 1, pp. 41-55, March 2019. doi: 10.1109/TNSM.2018.2889009

[20.] Kim, M. Park and D. H. Lee, "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection," in IEEE Access, vol. 8, pp. 70245-70261, 2020. doi: 10.1109/ACCESS.2020.2986882

[21.] Z. Lu, J. Xu, Y. Wu, T. Wang and T. Huang, "An Empirical Case Study on the Temporary File Smell in Dockerfiles," in IEEE Access, vol. 7, pp. 63650-63659, 2019. doi: 10.1109/ACCESS.2019.2905424

[22.] W. Zhang, B. Zhang, Y. Zhou, H. He and Z. Ding, "An IoT Honeynet Based on Multiport Honeypots for Capturing IoT Attacks," in IEEE Internet of Things Journal, vol. 7, no. 5, pp. 3991-3999, May 2020. doi: 10.1109/JIOT.2019.2956173

[23.] C. Tsung, H. Hsieh and C. Yang, "An Implementation of Scalable High Throughput Data Platform for Logging Semiconductor Testing Results," in IEEE Access, vol. 7, pp. 26497-26506, 2019. doi: 10.1109/ACCESS.2019.2901115

[24.] R. R. Karn, P. Kudva and I. A. M. Elfadel, "Dynamic Auto selection and Autotuning of Machine Learning Models for Cloud Network Analytics," in IEEE Transactions on Parallel and

Distributed Systems, vol. 30, no. 5, pp. 1052-1064, 1 May 2019. doi: 10.1109/TPDS.2018.28768.

[25.] S. Sultan, I. Ahmad and T. Dimitriou, "Container Security: Issues, Challenges, and the Road Ahead," in IEEE Access, vol. 7, pp. 52976-52996, 2019. doi: 10.1109/ACCESS.2019.2911732C. Tozzi. (Jan. 2017). What Do Containers have to Do With DevOps, Anyway?. Available: https://containerjournal.com/2017/01/11/containers-devops-anyway/

[26.] 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.

[27.] 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.

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

[29.] 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.

[30.] 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.

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

[32.] J. Soldani, D. A. Tamburri, and W.-J. Van Den Heuvel, ‘‘The pains and gains of microservices: A systematic grey literature review,’’ J. Syst. Softw., vol. 146, pp. 215–232, Dec. 2018.

[33.] 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.

[34.] R. Buyya et al. (2017). ‘‘A manifesto for future generation cloud computing: Research directions for the next decade.’’ [Online]. Available: https://arxiv.org/abs/1711.09123

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

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

[37.] 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.

[38.] 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.pdf

[39.] M. Pearce, S. Zeadally, and R. Hunt, ‘‘Virtualization: Issues, security threats, and solutions,’’ ACM Comput. Surv., vol. 45, no. 2, Feb. 2013, Art. no. 17.

[40.] M. Pearce, S. Zeadally, and R. Hunt, “Virtualization: Issues, security threats, and solutions,” ACM Computing Surveys (CSUR), vol. 45, no. 2, p. 17, 2013.

[41.] G. Pék, L. Buttyán, and B. Bencsáth, “A survey of security issues in hardware virtualization,” ACM Computing Surveys (CSUR), vol. 45, no. 3, p. 40, 2013.

[42.] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, “A survey on virtual machine migration: Challenges, techniques, and open issues,” IEEE Communications Surveys Tutorials, vol. 20, no. 2, pp. 1206–1243, Secondquarter 2018.

[43.] W. Gentzsch and B. Yenier, “Novel software containers for engineering and scientific simulations in the cloud,” International Journal of Grid and High Performance Computing (IJGHPC), vol. 8, no. 1, pp. 38–49, 2016.

[44.] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, “Performance evaluation of container-based virtualization for high performance computing environments,” in Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on. IEEE, 2013, pp. 233–240.

[45.] M. Stuart, “Evolving container architectures,” https://wikibon.com/evolving-container-architectures/, 2015, accessed: 13.11.2017.

[46.] R. Peinl, F. Holzschuher, and F. Pfitzer, “Docker cluster management for the cloud-survey results and own solution,” Journal of Grid Computing, vol. 14, no. 2, pp. 265–282, 2016.

[47.] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, “Microservices: yesterday, today, and tomorrow,” in Present and Ulterior Software Engineering. Springer, 2017, pp. 195–216.

[48.] B. Golden, “3 reasons why you should always run microservices apps in containers,” https://techbeacon.com/app-dev-testing/3-reasons-why-you-should-always-run-microservices-appscontainers, 2016, accessed: 11.11.2017.

[49.] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin et al., “Meltdown: Reading kernel memory from user space,” in 27th USENIX Security Symposium (USENIX Security 18), 2018, pp. 973–990.

[50.] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, “Spectre attacks: Exploiting speculative execution,” arXiv preprint arXiv:1801.01203, 2018.