Container Security: An Extensive Roadmap

Suganthi Subramanian[1,*] Prasad B Honnavalli[2] Shylaja S S[3]

[1,2,3] *Department of CSE, PES University, Bangalore, India.*
[*]*Corresponding author. Email: suganthis@pes.edu*

ABSTRACT
The containers play a crucial role in the cloud environment during application deployment as it shares same OSkernel. It reduces resource requirements and start-up time for deploying applications by an individual organizations or users. Even though containers provide light-weight virtualization, it generates a security bottleneck for the number of dedicated resources, libraries, and applications since the container isolationis comparatively weak to the legacy VMs. In the general architecture of container, attackers can perform privilege escalation by exploiting the kernel vulnerabilities to gain the root privilege andleaks the critical information of a system. To address the present security concerns in the container, a better security based solutionis essential. In this work, an extensive analysis is performed to predict the various existing access control mechanisms usedfor security purposes and the challenges encountered during the architecture modeling. Some use cases are considered to ensure the fulfilment of security requirements such as container protection, inter-container protection, and host protection, andit needs to provide both software and hardware solutions. This work also includes the research problems, research gaps, and further research extensions to provide security to the containers.

Keywords: *Applications, Attacks and Vulnerabilities, Container, Deployment, Lightweight, Privileges, Resource Utilization,Security.*

1. INTRODUCTION

A Virtual Machine (VM) in computing is an emulation ofa computer system. VM is based on the architecture of acomputer and offers physical computer behaviour. Specialized hardware, software, or a combination may be included in theirimplementations and it provides definitive safety. Typically, itdenotes various operating systems running concurrently on a computer system. System, library and other programmes may appear as if they are unique to the virtualized visitor system and not connected to under the host operating system for appsrunning on the virtualized maker.

Cloud computing (CC) is extensively espoused for consolidating various computational resources. The multi-tenancy concept facilitates the CC features to handle the computationalinstances from diverse tenants that work over the physical server [1]. The container cloud from the available cloud services emerges as a lightweight substitution for the traditional virtual machine (VM) based cloud infrastructures. Generally, containers define as an operating system (OS) based virtualization model with successive building blocks over the Linux kernel. It is composed of control techniques/resource isolation (cgroup and namespace) and security methods (seccomp, AppArmor, SELinux, and capabilities). With the elimination of added abstraction layers overhead, containers are competent toattain superior performance and outperform various VM basedsystems for certain factors [2].

Generally, the VM should hold its copy (own) of libraries, OS, applications, and dedicated resources. Thus, it shows some adverse or harmful effects on the storage size and performance,i.e., start-up time. This adverse effect over the micro services and software development practice in DevOps emphasizesa better solution than VM. It is not effective to run the micro services on an individual VM owing to the start-uptime and resource utilization [3]. The emerging function of container based virtualization is considered as the

lightweight substitution to the VMs. Containers can distribute the OSkernel indeed of duplication for all the VMs. It extensively diminishes the start-up time and the resources required for all images [4].

The emergence of micro service architectures assists in the increase of software suppleness where some software functionality turns into an independent scaling, deployment, versioning and development units. This architectural model is utilized by various organizations like Netflix, Spotify, Twitter,and Amazon for delivering the software [5]. Here, considers are depicted as the standardization model for deploying micro services and cloud applications. The containers are also essential for the future CC model.

The isolation between containers which are running on the same OS kernel by the host, can exploit kernel vulnerabilities to escalate their privileges and fully compromise the host (and all the other containers running on it). Therefore, securityis determined as the foremost obstacle to container adoption in a wider manner.

However, diverse reviews concentrate on addressing these issues, as the researchers do not concentrate on issues related to container security [6]. Container securityis divergent from VM complement as it is based on diverse architectural assistance. Therefore, there should be a better understanding of the security threats and the corresponding solutions. It is essential owing to the shortage of extensive analysis regarding the threat issues [7]. The solutions providedby some researchers are problematic due to the specific use case model. For instance, some trusted environment is utilizedto facilitate the functionality of the container on untrusted hosts. Therefore, tracking these use cases is considered to be more annoying for the researchers.

This survey provides an extensive analysis of the security measures and the analysis towards the use case model for the host-container level. There should be a better understanding of the security issues over the containers and the essential metrics to protect them. In addition, the users cannot avoid the access control functionality while downloading the container images. Some use cases need to ensure the security requirements for both hardware and software's and the factors for security analysis include container protection, inter-container protection, and host protection.

2. CONTAINERS: AN OVERVIEW

Various investigators give different names to specify containers which include lightweight virtualization and OS-level virtualization. Some of the examples of container managersare RKT, LXC, and Docker. Various investigations concentrate on Docker as it is the foremost essential container during the runtime environment. Hardware virtualization specifies the conventional hypervisors and VMs. Containers help to enhance the drawbacks identified over the VMs [8]. Initially, containers share the same resources and OS kernel, while the VMs need their copy. Then, the functionality of the containers is instantly started and stopped while VMs are the certain time for initiating the function. Also, it is known that containersare more efficient when compared to the VM for certain applications like micro services as containers are lightweight, and it does not need a complete OS copy for every image.

According to the observations from different study and underlying implementations, we can classify the containers into two categories namely Application container and System container. Containers are usually application centered which is a standalone, all-in-one package for a software application and it includes the binaries, software dependencies and the hardware requirements needed to run an application, all wrapped up into an independent, self-contained unit knownas application containers.

2.1. System and Application Containers

A system container is the oldest container type which behaves like a standalone system, which doesn't require a software or custom images such as Docker. System containersare similar to virtual machines (VMs) but with little overhead and easily manageable. It is most suitable for traditional or legacy monolithic applications, as they allow reusing architectures, tools, and configurations implemented for VMs. The functionality of the system container is represented in Fig 1.

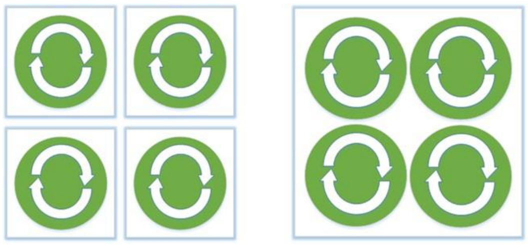
Figure 1 System and Application Container

An application container is referred as a standalone, self-contained unit. The functionality of the application containeris represented in Fig 1 and the only difference is that a system container runs on a base OS whereas application container is used for running an application inside the container and it is created using the Union File System service in image layers and the software technique, called Copy-on-Write (CoW), allowing a file system to look as writable, but without permittingwritten information to change the file system’s content. Onthe other end, a system container encapsulates an operating system, which is likely to be enhanced with new applications, libraries, and tools. Except for the very last one, which is a writable image, all images are read-only. This means that when the container is erased, the data in the container is lost.

Containers have security related limitations when it comes to security, and we need to understand those limitations before building an application. So in this paper a detailed research study is given towards analyzing the vulnerabilitiesin application container.

3. RELATED WORK

Moreover, containers require the complete OS kernel functionality by the host, which is shared among various containers. Also, the design of micro service emphasizes the significance of the transient state containers where data determination moves to other data services. It is also determined as the standard form for deploying cloud micro services [9]. A novel delivery model known as container as a Service (CaaS) is created for CC and the various companies move forwarded to offer container services as it facilitates a wide variety of container based applications over the real-time markets. Moreover, OS-level virtualization is a promising solution with diverse advantages, i.e., OS kernel (host) shared diverse security issues, making them less secure towards the VMs. Danev *et al.* [10] provide an extensive analysis of the most accessible container management model. The author intends to provide various industrial and academic requirements and maps them to meet the requirements based on the provided case study. Also, the author predicts the gaps among the tools and the requirements that need to be fulfilled. The provided tools need to overcome certain drawbacks.

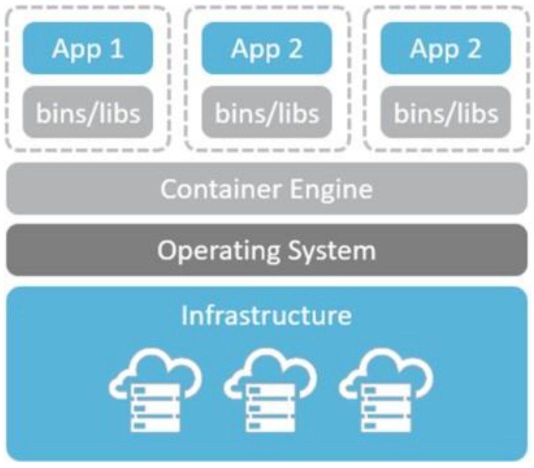

Figure 2 Container based Cloud

Wan *et al.* [11] discuss certain monolithic applications needed for the software to be coupled and avoid independent execution. However, monolithic applications function over the container, and it is highly essential to make use of a micro service architectural model with these containers. With certain special kinds of micro services and Service Oriented Architectures, some applications are considered to be monolithic. Subsequently, micro services assist in constructing certain applications, which are composed of loosely coupled parts that work independently. Similarly, the micro service architectures are revolutionized for the construction of certain applications. It makes the developer design more innovative and newer technological models [12]. Based on this, itis known that containers and micro services are probably related to one another and provides standard deployment for micro services. The functionality of the micro services over the separate VMs is not as efficient as VMS are slightly heavier compared to the container model. The containers are essential and work as an alternative to the VMs with huge benefits [13]. The significance of the containers emphasizes the significance of the micro service architectures compared to the conventional monolithic architectural model. However, the containers encounter various security issues which areconsidered as the barrier. Fig 2 shows the container based cloud model.

4. REVIEW ON SECURITY ISSUES OVER THE CONTAINERS

The developers performed certain experimentation over the Linux machine using the LXC, and Docker containers with the default configuration where the containers are provided with essential privileges (from users’ perspective) like commercial containers. The Linux machine is provided with two diverse kinds of interfaces, i.e., memory based file systems and systemcalls. The latter model is specifically designed for user processors for requesting the kernel services and shows backward compatibility. Similarly, the former model is more feasible with the kernel functionality, kernel parameter adjustment, and kernel data accessing mechanism [14]. It enables kernel data manipulation through I/O operations. Linux machine possesses a huge number of memory based files system for kernel operations. Some cross validation tools are modeled forautomatically predicting the memory based files that project the information to the containers. The preliminary concept for exploring the pseudo files recursively is sysfs and procfs. The files are reordered based on the file paths and perform certain pairwise analyses over the file contents [15]. The resources are accessed by any specific pseudo files which is not names pace over the Linux kernel. The containers can retrieve the own data when it is properly namespaced. With the availabilityof the cross-validation tools, the pseudo files can be identifiedpre-dominantly and exposed to the host information of the container. The pseudo files are accountable for host information leakage where the leakage channel possesses various host information factors. For instance, the containers can acquire hardware based sensor data like power consumption, DRAM, and temperature via the temperature sensors. The utilizationof disk I/O, memory and processors are exposed towards the containers. The information leakage is more harmful, and various adversaries exploit it during the attack launch [16]. The preliminary cause of information leakage is analyzed with the analysis over the kernel code. These leakages are encountered due to the incomplete execution of namespaces over the kernel. The cause for this leakage issue is listedbelow: Linux sub-system is not completely namespace, andthe context validation is missing for the prevailing namespace.Some general security requirements are given in Table I.

Table 1. Security Requirements

S. No	Requirements	Explanation
1	System integrity	The information over the system model needs to be preserved from unauthorized variations over the content (malware)
2	System protection	The transmission of information over the network channel needs to be protected and monitored
3	Service acquisition	More feasible and reliable services need to be adopted. With the adoption of external services, the information transmission is done in an independent manner
4	Maintenance	Security measure should be given in a cost efficient manner
5	Authentication and identification	The user information accessing system and services need to be verified and validated.
6	Access control mechanism	Information transmission, modification, unauthorized access have to be avoided.

5. SECURITY THREAT TOWARDS THE CONTAINER IMAGE

The container engine needs to preserve the container's execution environment. The container utilizes cgroups and namespaces to apply resources, variables, and application management during container functionality to fulfill the environmental operations. However, the container engine fails to preserve the container images; but preserves the application with the container images. It does not preserve the container image needed for running the application. In some environments, unauthorized users can access the container image directly to vary the files or sometimes leak the essential information. Specifically, in the container platform, various container images are executed where the modification overthe container layers is reflected in the container applications [17]. The applications that work over the container platform need to run over the secured environment. The container image execution has to fulfill the process of isolation with the namespace. Independent analysis needs to fulfill the resource allocation for certain applications in the containers using cgroups. The container platform assists some unauthorized modification over the union file system. It is made of multiple layers and treated as a readable and writable file system. Some unauthorized individuals are accessed directly over the container image to modify the files or leakage information [17]. SecureOS is utilized to enhance security vulnerabilities as the present model does not determine the container platform. Also, it does not enhance the security vulnerability over the container platform.

The container image comprises multiple layers where the stateis measured as a secured environment when the container is executed while the container image is preserved. However, the container platform does not require protection over the container images. There is no proper security measure to preserve the container images. Thus, the container platformis not so secure for preserving the container images [18]. The objects need to be preserved with an access control mechanism for essential information using secureOS. The container image directory is performed over the image layer, whichis dynamically generated while downloading the images. Itis impossible to predict the container image as it is SHA-256 type. The ultimate target of the container platform isto operate, execute, and distribute the applications need to offer information services. It is accountable for preservingthe container images based on security measures. To offer a better security environment, the container image based access control is anticipated to block unauthorized access to containerimages. The access control mechanism for container images is isolated from the container engine to fulfill the resources needed for container functionality [19]. The container image based layer directory needs to be preserved dynamically, and authorized containers need to access the layer directory of the container image. The policy identifier gathers the essential userinformation that accesses the container image of the host system. The kernel policies deals with the addition and deletionof access control policies and the policy table is a database andperforms access control policies for registration purpose. The access control policies are composed of user ID, containerID, and container based image layer directory. Finally, user access control handling rejects unauthorized access towards the container image layer by examining the I/O of the user’s file [20]. The event over the container image accessing isperformed using the container platform, and another kind of access is rejected. Table II depicts the threat scenario for container protection.

Table 2. Threat Scenario for Container Protection

Threat	Attacks	Probable environment	Outcomes
Insecure runtime	Execution of remote code	Remote code execution over the Docker. Attacks lead to a configuration error in runtime environment	The runtime containers have to be monitored for scanning vulnerabilities
Boundless network access	Vulnerability scanning	A compromised container needs access to the network. Can identify vulnerability by performing vulnerability scanning over the container	Containers need to be separated as virtual networks w.r.t. its sensitivity nature
Inter-container traffic	ARP spoofing	Traffic over the containers leads to compromised containers and causes like DoS; leading to flooding and performance degradation	Establish efficient communication when needed, efficient algorithms to overcome the attacks like DoS over the Docker system
Vulnerabilities	Attacks on containers	Vulnerable applications may cause DoS attacks over the available containers and utilize many available resources affecting the functionality of the container	The image scanning must be performed periodically over the application
Untrusted image environment	Attacks over the container in the host environment	Untrusted image over the containers leads to serious threats and poses pre-installed vulnerability which scans the network images to predict vulnerability	Only trusted images need to be validated using the proper security based signature, i.e., scanning the vulnerability

5.1. Case study 1

The application that runs over the container needs to be more trustful or semi-trustful, or vulnerable where the application does not have any access control. Some applicationsact as the root access environment. The applications needto capture the control over the container manager when the container has the competency to target other containers and host systems. Sometimes, the malicious container takes control over the available containers. The ultimate goal is to reduce vulnerabilities.

5.2. Case study 2

When the container is semi trustful or vulnerable towards the host, it is specified as CM—1, where CM specifies a vulnerable or semi trusted container. This semi trusted container needs to access the host information confidentiality or pretends container image needed for running the application. In some environments, unauthorized users can access the container image directly to vary the files or sometimes leak the essential information. Specifically, in the container platform, various container images are executed where the modification overthe container layers is reflected in the container applications [17]. The applications that work over the container platform need to run over the secured environment. The container image execution has to fulfill the process of isolation with the namespace. Independent analysis needs to fulfill the resource allocation for certain applications in the containers using cgroups. The container platform assists some unauthorizedmodification over the union file system. It is made of multiple layers and treated as a readable and writable file system. Some unauthorized individuals are accessed directly over the container image to modify the files or leakage information [17]. SecureOS is utilized to enhance security vulnerabilities asthe present model does not determine the container platform. Also, it does not enhance the security vulnerability over the container platform.

5.3. Case study 3

Some malicious or untrusted containers are placed over thehost or other hosts. Here, the attackers may take control over other applications. However, the container needs to maintain honesty as it is not related to the container specific problem. The untrusted container may take control over the confidentialdata of other containers, consume the resources, and spoil the integrity of the applications. Moreover, the container needsto carry out attacks and verifies container availability. The malicious container sucks the resources of the host. The targetis to maintain the containers from one another. The significant factor is to preserve other containers without damaging other containers and spoils the container information. Sometimes, the high risk attacks may influence the containers and mount the attack over OpenVZ, Docker, and LXC. The launches of attacks lead the adversaries to leak sensitive information from other containers and host OS. It shows a drastic effect on the performance of cloud services. The malicious container takes control over the host system and sucks the memory. More specifically, Meltdown and spectre are the two complex attacksthat show severe consequences over the containers.

5.4. Case study 4

The untrusted host needs to be avoided, and the containers need to be more feasible and robust. However, here the container acts as a malicious or semi trusted container. These untrustful containers intend to learn the sensitive and confidential information of the containers as it manages other network devices like processors, storage, memory, and work devices. The malicious host pretends to capture the containers' integrity, i.e., the launch of active/passive attacks makes the container vulnerable or malicious containers. Some of these attack types include unauthorized data access and profiling container activities. Active attacks are more harmful than passive attacks as the malicious host can manipulate the application's characteristics.

6. REVIEWS ON PROTECTION MECHANSIMS

There are two diverse types of protection mechanisms against the containers. They are hardware and software protections (See Fig 3). Initially, namespace performs isolation and system virtualization for performing the process. It works as an identifier that links the isolated instances and other global resources [21]. Namespace segments the hostname, user, process, file system, and other related components. Ithas to validate the isolation process and establish bindingamong the other available containers. The major issue related to the namespace is the resource unawareness towards thenamespace over the devices. There is a huge namespace thatis accountable for the isolation of resources.

Namespace: PID namespace

The above instance specifies that the process needs to execute its namespace. Next is the establishment of inter-containerprotection with namespaces, a powerful feature of Linux for the isolation of resources among the available containers. It assists in the elimination of containers from being accessed from the resources and provides higher security measures. The containers can overview the containers' processes and communicate with one another.

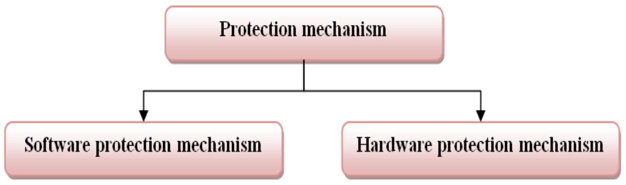

Figure 3 Protection Mechanisms

Thus, PID namespace is applied to containers for isolating various resources. The third method is the protection of the host using namespace using the containers. The compromised containers have to escalate the root cause of the host process and disruptive operations. This namespace process helps to mitigate the attack risks, and in some cases, the containers escape from the attacks over the Docker. The information leakage channels over the containersleak the sensitive information and facilitate the adversariesto launch attacks over the CSP. Some investigators foundthat power based namespace is the only solution against the container level resource consumption.

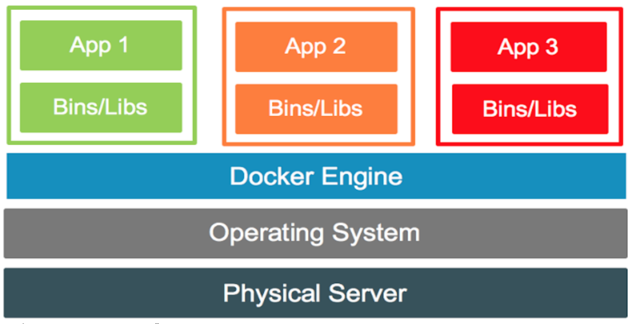

Figure 4 Docker Engine

Control groups (CGroups) are features that are accountable for resource utilization like I/O process, system memory, CPU, and network bandwidth. The major difference betweennamespace and CGroups are given below. The former modelneeds to manage the container resources, while the lattermodel needs to deal with the number of resources usedfor analysis. CGroups are essential for protecting the inter-container and facilitates resource utilization [22]. Therefore,the container may not use more resources even in case ofresource availability. It helps to preserve the containers fromvarious attacks like DoS. The container must use the availablehost RAM than other containers that are not operated properly. Some real-time protection methods like container drones are used where CPU protection uses CGroups for allocating the setof tasks. The rising priority concepts are restricted towards theDocker features. MemGuard kernel is used for protecting the memory from DOS attack and prevent CPU core from memoryaccess [22]. The next protection mechanism is the securecomputation model, where the kernel feature filter outs thesystem calls towards the kernel. It fine grains the capabilities and assists in reducing the number of system calls from the containers. It reduces the threats, and the leverage kernel may exploit the system call. Some sets of container namespaces are generated automatically using Docker to establish a secureenvironment. Sometimes, Docker depends on CGroups forprocessing the container runtime (See Figure 4). It considers some metrics like I/O and CPU utilization where the resource configurations are either hard or soft constraint [23]. The former model uses a certain amount of resources towards the container, while the latter model provides the container with essential resources over the machine.

Table 3. Total No.of.Vulnerabilities Reported by Clair tool for popular 10 Docker images

Image name	No.of.downloads(in million)	Total	High	Medium	Low	Negligible	Unkn-own	Dos-attack	Memory-corruption	Man-in-the-middle-attack
ubuntu	1000	19	0	2	13	4	0	yes	yes	no
couchbase	50	21	0	3	14	4	0	yes	yes	no
percona	50	36	8	25	3	0	0	yes	yes	no
redis	1000	61	2	7	13	39	0	yes	yes	yes
Amazon linux	100	1	0	1	0	0	0	no	yes	no
mariadb	1000	23	0	4	15	4	0	yes	yes	no
mysql	1000	73	2	7	13	49	2	yes	yes	yes
mongo	1000	41	0	13	19	9	0	yes	yes	no
rabbitmq	1000	20	0	3	12	4	0	yes	yes	no
cassandra	100	24	0	3	16	4	0	yes	yes	no

Trusted Platform Modules (TPM) is used as a crypto-graphic processor when coming to hardware based protection mechanisms. It offers hardware support for accelerating the provided algorithm, sealing data, and secure boot. With the adverse effect of virtualization and cloud computing process, the researchers need to check with the alternatives towardsthe hardware TPM to offer appropriate hypervisors. It checks the availability of the TPM to the VMs plethora. It is also known as virtual TPM generated for matching the container requirements. The container manner needs to check with the OS kernel (host) for creating some vTPM and allocate it towards the new container. Moreover, the protection level is comparatively lesser than the full hypervisors [24-28]. There are two diverse probable methods to fulfil the security requirements. Trust has to be established among the root of TPM. Therefore, the host OS considers the trust and produces TPM in a trusted manner. Therefore, the containers need to use hash for establishing the feature execution. TPM needs to launch trust among the host as it offers its instances for deploying the protocols for allocating the endorsement keys [29-32].

7. DISCUSSION

The containers rely on kernel sharing where the malicious container may leak the sensitive information over the host, which causes severe threat conditions over the CSP. It has to be addressed as the threat may contaminate the functionality ofthe container and leads to huge damage. Standard mechanisms for container deployment, evaluation, and communication protocol are essential to fulfil the simplicity, usability, automation, and ease of use. In some cases, digital forensics is utilizedfor examining security measures [25]. It has to verify thatthe current approaches are appropriate for the containers. Some investigators found that docker images hold higher risk vulnerability and lead to various complications. Thus, efficient vulnerability assessment tools are required for docker images [33].

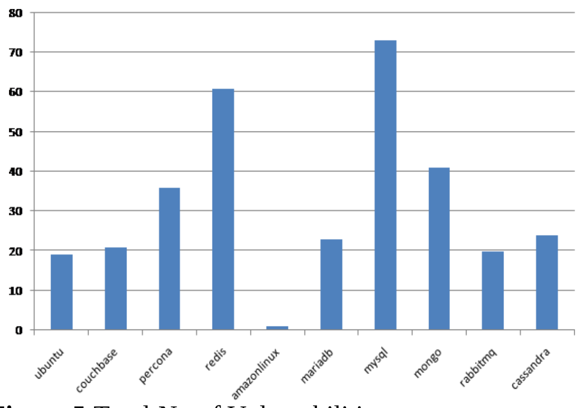

Figure 5 Total No.of.Vulnerabilities

Based on the survey analysis we used Clair, an open source container scanning tool to scan the popular 10 images from Docker repository. Table 3 shows the initial experimental result to identify the vulnerabilities. Furthermore, the work will be extended by analyzing the images with various open sourcetools and to develop a better container vulnerability detection approach. The various methods of static analysis detect the vulnerabilities only in the docker images but still the metadataof the images like package name, package information needsto be extracted. These all factors should be considered along with the code inspection so that the complete vulnerabilities will be identified to make container a more secure [34].

The total no. of. vulnerabilities are identified in 10 Docker image's project (as shown in figure 5). This figure shows a bar chart with Docker images name on x-axis, and total number of vulnerabilities detected on the y-axis. There are in total 19 vulnerabilities detected in ubuntu image. Out of these 19 vulnerabilities,2 are medium,13 are low. But, most of the vulnerabilities detected are due to DOS attack and Memory corruption or buffer over-read attack. Some strong and secure methods should be imposed to overcome these attacks [35-39].

8. CONCLUSION

From the extensive review, it is known that the containers are essential for CC, and it is evolving technology. In modern era, it is widely used in cloud environments which are determined as the emerging field of research. The primary barrier that reduces the widespread of containers is security issues. Thus, the security needs to be fulfilled to establish better solutions as there are very few research works that concentrate on security factors.

The main idea of the proposed work largely focuses in improving the isolation of containers by providing an automated hardening policy based on (a) static analysis for the execution metrics of container images (b) dynamic analysis by continuously monitoring the activity of the container in runtime. Also to enforce the Mandatory Access Control(MAC) policies to all the critical components of the container images so that unintended operations can be prevented both in host and container.

REFERENCES

[1] 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.

[2] Iera, and T. Taleb, "Evaluating performance of containerized IoT ser- vices for clustered devices at the

network edge", IEEE Internet Things J., vol. 4, no. 4, pp. 1019 1030, Aug. 2017.

[3] Jamshidi, C. Pahl, N. C. Mendonc¸a, J. Lewis, and S. Tilkov, "Microser- vices: The journey so far and challenges ahead", IEEE Softw., vol. 35, no. 3, pp. 24 35, May/Jun. 2018.

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

[5] Gentzsch and B. Yenier, "Novel software containers for engineering and scienti c simulations in the cloud", Int. J. Grid High Perform. Comput. (IJGHPC), vol. 8, no. 1, pp. 38 49, Jan. 2016.

[6] Merkel, "Docker: Lightweight Linux containers for consistent develop- ment and deployment", Linux J., vol. 2014, no. 239, Mar. 2014

[7] ao, Z. Gu, M. Kayaalp, D. Pendarakis, and H. Wang, "Container- Leaks Emerging security threats of information leakages in container clouds", in Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN), Jun. 2017, pp. 237 248.

[8] Zhang, and X. Zheng, "A survey on security issues in services com- munication of Microservices-enabled fog applications", in Concurrency and Computation: Practice and Experience. Hoboken, NJ, USA: Wiley, 2018

[9] Stumpf and C. Eckert, "Enhancing trusted platform modules with hardware based virtualization techniques", in Proc. 2nd Int. Conf. Emerg. Secure. Inf., Syst.Technol., Aug. 2008, pp. 1 9.

[10] anev, R. J. Masti, G. O. Karame, and S. Capkun, "Enabling secure VM-vTPM migration in private clouds", in Proc. 27th Annu. Comput. Secure. Appl. Conf., Dec. 2011, pp. 187 196.

[11] Wan, Z. Xiao, and Y. Ren, "Building trust into cloud computing using virtualization of TPM", in Proc. 4th Int. Conf. Multimedia Inf. Netw. Secure., Nov. 2012, pp. 59 63.

[12] Mobius, W. Dargie, and A. Schill. "Power Consumption Estimation Models for Processors, Virtual Machines, and Servers". IEEE Trans- actions on Parallel and Distributed Systems, 2014.

[13] Wu, Z. Xu, and H. Wang. "Whispers in the Hyper- space: Highspeed Covert Channel Attacks in the Cloud". In USENIX Security, 2012.

[14] Xu and D. Marinov, "Mining container image repositories for software con guration and beyond", in Proc. 40th Int. Conf. Softw. Eng. New Ideas Emerge. Results, 2018, pp. 49-52.

[15] Lu, J. Xu, Y. Wu, T. Wang, and T. Huang, "An empirical case study on the temporary le smell in docker les", IEEE Access, vol. 7, pp. 63650 63659, 2019.

[16] Holger Gantikow, Christoph Reich, Martin Knahl and Nathan Clarke,"Rule based Security Monitoring of ContainerizedWorkload", in Pro- ceedings of the 9th International Conference on Cloud Computing and Services Science, 2019.

[17] Anwar, S. Potter, C. Narayanaswami, W. Yurick, C. A. Gunter, and R. H. Campbell, "Detecting and mitigating denial-of-service attacks on voice over IP networks," IBM Watson Res., Columbia Univ., Sep. 2008.

[18] Chen, N. Li, and M. Z., "Analyzing and comparing the protection quality of security-enhanced operating systems", in Proc. NDSS, Feb. 2009, pp. 11-16.

[19] Kaewkasi and K. Chuenmuneewong, "Improvement of container scheduling for docker using ant colony optimization", in Proc. 9th Int. Conf. Knowl. Smart Technol. (KST), Feb. 2017, pp. 254 259.

[20] Kehrer, F. Riebandt, and W. Blochinger, "Container based module isolation for cloud services," in Proc. IEEE Int. Conf. Service-Oriented Syst. Eng. (SOSE), Apr. 2019, pp. 17704 -17709.

[21] Polemi and T. Ntouskas, "Open issues and proposals in it security management of commercial ports: The s-port national case", in Proc. 27th Inf. Secure. Privacy Conf., G. Crete, Ed., 2012, pp. 567 572.

[22] Ahmed and G. Pierre, "Docker container deployment in fog computing infrastructures", in Proc. IEEE Int. Conf. Edge Comput. (EDGE), Jul. 2018, pp. 1 8.

[23] Thalheim, P. Bhatotia, P. Fonseca, and B. Kasikci, "Cntr: Lightweight OS Containers", in Proc. USENIX Annu. Tech. Conf., Jul. 2018, pp. 199-212.

[24] Han, H. K. Lee, G. Y. Gim, and S. J. Kim, "Empirical study on anti- virus architecture for container platforms", IEEE Access, vol. 8, pp. 134940 134949, 2020.

[25] Raho, A. Spyridakis, M. Paolino, and D. Raho, "KVM, Xen, and docker: A performance analysis for ARM based NFV and cloud computing", in Proc. IEEE 3rd Workshop Adv. Inf., Electron. Electr. Eng. (AIEEE), Nov. 2015, pp. 1-8.

[26] Bhuvaneswary, N., S. Prabu, S. Karthikeyan, R. Kathirvel, and T. Saraswathi. "Low Power Reversible Parallel and Serial Binary Adder/Subtractor." Further Advances in Internet of Things in Biomedical and Cyber Physical Systems (2021): 151.

[27] Bhuvaneswary, N., S. Prabu, K. Tamilselvan, and K. G. Parthiban. "Efficient Implementation of Multiply Accumulate Operation Unit Using an Interlaced Partition Multiplier." Journal of Computational and Theoretical Nanoscience 18, no. 4 (2021): 1321-1326.

[28] Le, Ngoc Tuyen, Jing-Wein Wang, Duc Huy Le, Chih-Chiang Wang, and Tu N. Nguyen. "Fingerprint enhancement based on tensor of wavelet subbands for classification." IEEE Access 8 (2020): 6602-6615.

[29] Naeem, Muhammad Ali, Tu N. Nguyen, Rashid Ali, Korhan Cengiz, Yahui Meng, and Tahir Khurshaid. "Hybrid Cache Management in IoT based Named Data Networking." IEEE Internet of Things Journal (2021).

[30] Pham, Dung V., Giang L. Nguyen, Tu N. Nguyen, Canh V. Pham, and Anh V. Nguyen. "Multi-topic misinformation blocking with budget constraint on online social networks." IEEE Access 8 (2020): 78879-78889.

[31] Subramani, Prabu, K. Srinivas, R. Sujatha, and B. D. Parameshachari. "Prediction of muscular paralysis disease based on hybrid feature extraction with machine learning technique for COVID-19 and post-COVID-19 patients." Personal and Ubiquitous Computing (2021): 1-14

[32] Rajendrakumar, Shiny, and V. K. Parvati. "Automation of irrigation system through embedded computing technology." In Proceedings of the 3rd International Conference on Cryptography, Security and Privacy, pp. 289-293. 2019.

[33] Puttamadappa, C., and B. D. Parameshachari. "Demand side management of small scale loads in a smart grid using glow-worm swarm optimization technique." Microprocessors and Microsystems 71 (2019): 102886.

[34] Z. Guo, L. Tang, T. Guo, K. Yu, M. Alazab, A. Shalaginov, "Deep Graph Neural Network based Spammer Detection Under the Perspective of Heterogeneous Cyberspace", Future Generation Computer Systems, https://doi.org/10.1016/j.future.2020.11.028.

[35] Y. Sun, J. Liu, K. Yu, M. Alazab, K. Lin, "PMRSS: Privacy-preserving Medical Record Searching Scheme for Intelligent Diagnosis in IoT Healthcare", IEEE Transactions on Industrial Informatics, doi: 10.1109/TII.2021.3070544.

[36] N. Shi, L. Tan, W. Li, X. Qi, K. Yu, "A Blockchain-Empowered AAA Scheme in the Large-Scale HetNet", Digital Communications and Networks, https://doi.org/10.1016/j.dcan.2020.10.002.

[37] C. Feng *et al.*, "Efficient and Secure Data Sharing for 5G Flying Drones: A Blockchain-Enabled Approach," IEEE Network, vol. 35, no. 1, pp. 130-137, January/February 2021, doi: 10.1109/MNET.011.2000223.

[38] L. Tan, H. Xiao, K. Yu, M. Aloqaily, Y. Jararweh, "A Blockchain-empowered Crowdsourcing System for 5G-enabled Smart Cities", Computer Standards & Interfaces, https://doi.org/10.1016/j.csi.2021.103517

[39] K. Yu, L. Tan, X. Shang, J. Huang, G. Srivastava and P. Chatterjee, "Efficient and Privacy-Preserving Medical Research Support Platform Against COVID-19: A Blockchain based Approach", IEEE Consumer Electronics Magazine, doi: 10.1109/MCE.2020.3035520.