International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

Security Audit of Kubernetes based Container Deployments: A Comprehensive Review

Mayank Agrawal[1], Kumar Abhijeet[2], Smitha G.R.[3], Chethana R Murthy[4]

[1]*Dept. of Information Science and Engineering, RV College of Engineering, Bangalore*
[2]*Dept. of Information Science and Engineering, RV College of Engineering, Bangalore*
[3-4]*Professor, Dept. of Information Science and Engineering, RV College of Engineering, Bangalore, India*

Abstract - *In the age of open-source software, new and improved frameworks that are crowd worked to perfection are often set as an industry standard. Kubernetes, also an open-source software that took the DevOps community by surprise and became the new standard for deployments, is a product of this standard. In this paper, we have identified the various security vulnerabilities related to a Kubernetes based container deployments and laid out the best practices to inculcate while setting up the infrastructure for deployment using a set of guidelines to enable enterprises running cloud-based systems without worrying about the attacks for ransomware, crypto mining, data stealing and service disruption.*

Key Words: Cloud, Kubernetes, DevOps, Container, Docker, Deployment, Infrastructure

1. INTRODUCTION

In recent times, infrastructure development philosophy has moved to a DevOps based one, which combines both the realms for Development and Operations [1], hence using the nuances of programming and tricks of operation to create the infrastructure for an organization. After the advent of Kubernetes in the past decades, teams have rapidly moved from manual cluster deployments and maintenance to Kubernetes based orchestration.

Kubernetes is an open-source, portable, orchestration tool built upon the learnings of Google for managing microservices or containerized applications through a distributed cluster of nodes which facilitates both declarative configuration and automation. Kubernetes provides a highly resilient infrastructure with zero downtime deployment capabilities, automatic rollback, scaling, and self-healing of containers (which consists of auto-placement, auto-restart, auto-replication [2], and scaling of containers based on CPU usage). The main objective of Kubernetes is to hide the complexity of managing a fleet of containers by providing APIs for the required functionalities [3]. Kubernetes is portable in nature, meaning it can run on various public or private cloud platforms such as AWS, Azure, OpenStack, or Apache Mesos. It can also run on bare metal machines.

Discussing more about the components and architecture of a Kubernetes based system will bring more clarity on further discussions on its security. Kubernetes follows a client-server architecture. It's possible to have a multi-master setup (for high availability), but by default, there is a single master server which acts as a controlling node and immediate point of contact. The master server consists of various components including a Kube-apiserver, an etcd storage, a Kube-controller-manager, a cloud-controller-manager, a Kube-scheduler, and a DNS server for Kubernetes services. Node components include kubelet and Kub-proxy on top of Docker (a platform as a service to deliver software in packages called containers).

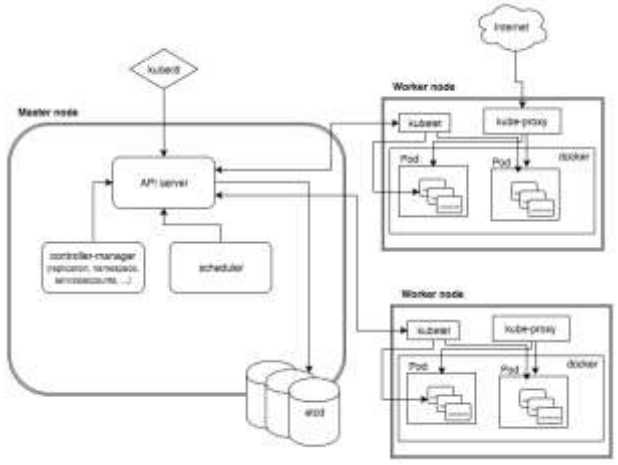

Fig. 1 High-Level Kubernetes Architecture Diagram

Making use of Kubernetes requires understanding the different abstractions it uses to represent the state of the system [4], such as services, pods, volumes, namespaces, and deployments.

Pod – generally refers to one or more containers that should be controlled as a single application. A pod encapsulates device modules, computing space, an eccentric network ID, and other module settings on how to run the containers.

Service - pods are volatile, that is Kubernetes does not guarantee that a given physical pod will be kept alive (for instance, the replication controller might kill and start a

new set of pods). Instead, a service represents a logical set of pods and acts as a gateway, allowing (client) pods to send requests to the service without needing to keep track of which physical pods actually make up the service.

Volume - Like a container volume in Docker, but a Kubernetes volume applies to a whole pod and is mounted on all containers in the pod. It is like a directory accessible to all the containers running in a pod. Kubernetes guarantees data to be preserved in volumes across container restarts. The volume will be removed only when the pod gets destroyed. Also, a pod can have multiple volumes (possibly of different types) associated.

Namespace - a virtual cluster (a single physical cluster may operate several virtual clusters) built for environments with a wide number of users distributed through several teams or ventures, to address problems. Objects inside a namespace must be different, so they cannot access objects in another namespace. A resource limit can often be assigned to a namespace to prevent using more than its share of the physical cluster.

Deployment - describes the desired state of a pod or a replica set, in a .yaml file. The Deployment Controller then changes the environment slowly (for example, adding or removing replicas) until the current state matches the desired state defined in the Deployment script. For example, if the .yaml file defines 2 replicas for a pod but only one is currently running, then an additional replica will automatically get created.

In the context of the Kubernetes concepts, container deployment is a method for quickly building and releasing complex applications. Kubernetes container deployment is a popular technology that gives developers the ability to construct application environments with speed at scale. Docker and Openshift are also widely accepted container deployment technologies.

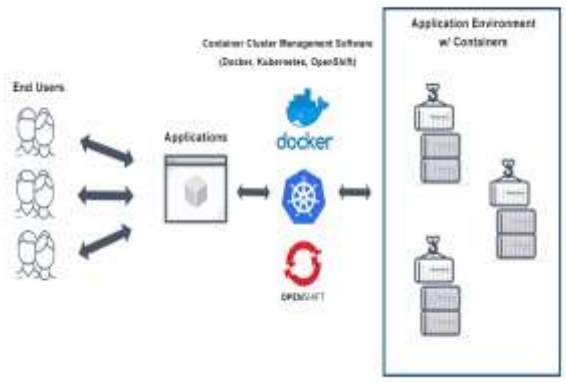

Fig. 2, Container Deployments

2. SECURITY CHALLENGES

Containers and frameworks such as Kubernetes enable businesses to automate many facets of application deployment, bringing enormous business benefits. But such latest architectures are as susceptible as conventional systems to attacks and hacks by hackers and outsiders, rendering Kubernetes protection a vital component for all architectures. Attacks on malware, crypto mining, code theft, and device interruption will start toward modern virtualized container-based systems in both private and public clouds [5].

To make matters worse, emerging techniques and innovations such as Docker and Kubernetes would be under assault to find alternatives to control the critical properties of an organization. The latest exploit in Kubernetes at Tesla is only the first of several container-based vulnerabilities that we will expect in the coming months and years.

The dynamic nature of containers creates the following Kubernetes security challenges:

- **Explosion of East-West Traffic** - Containers might be seamlessly deployed through hosts or perhaps even clouds, increasing the east-west or internal traffic that needs to be monitored for attacks significantly.
- **Increased Attack Surface** - Every container can have a specific surface area of attack and weaknesses that can be exploited. Additionally, attention should be extended to the additional attack surface introduced by container orchestration platforms such as Kubernetes and Docker.
- **Automating Security to Keep Pace** - Old protection models and software won't be able to keep up in an ever-changing container environment.

There are some vulnerabilities and attack vectors on Kubernetes which should be explored before moving on to remedial strategies and best practices [6]. Attacks on Kubernetes containers operating in pods may come from outsiders externally or internally across the network, including victims of phishing attacks whose networks are conduits for insider attacks. Here are a few examples:

- **Container compromise** - A misconfiguration or malfunction in a program helps the intruder to get into a container and start searching for network vulnerabilities, process controls, or file system.
- **Unauthorized connections between pods** - Compromised containers may attempt to link with the same hosts or with other operating pods to probe or launch an attack. Whilst the Layer 3 network controls whitelisting pod IP addresses may provide certain security, attacks on trustworthy IP addresses can only be identified through filtering of the Layer 7 network.

- **Data exfiltration from a pod** - Data theft is sometimes achieved using a variety of strategies that could involve a reverse shell in a pod connecting to a command/control system and network rerouting to hide sensitive data.

In addition to these vulnerabilities, attackers can also launch a series of malicious activities that can be considered as a "Kill Chain". These are the most damaging activities which together achieve the goal of attackers. These attacks can occur within a span of seconds or spread over days or even last for months.

Detecting events in a kill chain requires rigorous monitoring. Since the attack compromises multiple resources instances in the system, multiple layers of security monitoring are required. Some of the most critical vectors which have a high rate of chance of detection includes:

- **Network detection.** The network provides the starting gateway to attack. A malicious user typically enters via a network connection and uses subsequent opportunities to detect lateral movement.
- **Container monitoring.** An exploit within a system can be determined by detecting any unregulated system call, file system activity. A typical container can be monitored to determine to find any suspicious process or attempts being made to break out of the container. Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

3. KUBERNETES DEPLOYMENT BEST PRACTICES

The best practices and guidelines are laid out as follows which should be followed in order to prevent the Kubernetes infrastructure from the above attacks and vulnerabilities [8].

3.1 Preparing Kubernetes Worker Nodes for Production

The host networks for the Kubernetes worker nodes should be locked off before deploying any framework containers. The following are the methods to lock down the worker nodes.

- **Use namespaces** - Namespaces divide the Kubernetes infrastructure to different zones with non-stretchable boundaries.
- **Restrict Linux capabilities** - Linux offers a lot of services to change and modify network management, which can be used for attacks.
- **Enable SELinux** - Use this Linux distro to get better control over network systems.
- **Use a minimal Host OS** - It is advised to use an OS for the host system which satisfies only the required needs.
- **Update system patches** - Keep the system up-to-date is extremely crucial when it comes to infrastructure management as developers issue new patches every release cycle which solves a certain security vulnerability.
- **Run CIS Benchmark security tests** - The CIS benchmark tests ensure that the system is optimized to support the Kubernetes containers and deployments.

3.2 Real-time Kubernetes Security

When containers are in production, the three main security vectors for their protection are network filtering, container inspection, and host security.

3.2.1 Inspect and Secure the Network

A container firewall is a modern form of security service that extends conventional network protection strategies to the current Kubernetes cloud-native framework. There are numerous ways to protect a container network with a firewall including:

- Layer 3/4 filtering based on IP addresses and ports. This solution involves Kubernetes' network policy for continuous upgrading of regulations, securing applications when they shift and expand. Simple network optimization laws are not intended to include the rigorous control, reporting, and threat identification required for business-sensitive container deployments but can provide some security against unwanted connections.
- Web application firewall (WAF) attack protection may secure containers facing the web (typically HTTP-based applications) using methods that identify specific threats, comparable to web application firewalls features. However, the protection is restricted to external HTTP attacks and lacks the multi-protocol filtering that is often required for internal traffic.
- Layer-7 container firewall A container firewall with Layer 7 filtering and deep inter-pod traffic

monitoring of packets secures containers using network device protocols. Security is focused on the whitelists of client protocols as well as automated identification of common network-related device attacks such as DDoS, DNS, and SQL injection. Container firewalls are often in a unique role to integrate container process control and host protection in the surveyed threat vectors.

Owing to the complex design of the containers and the Kubernetes networking paradigm, standard methods cannot be used for network detection, forensics, and review. Easy activities like packet capture to monitor programs or to examine security incidents are no longer basic. New Kubernetes and container-conscious resources are required to perform protection, inspection, and forensic tasks on the network.

3.2.2 Container Inspection

Attacks also allow the use of power escalations and manipulative mechanisms to carry out or distribute an attack over the number of containers. Exploits of Linux kernel vulnerabilities (such as Dirty Cow), modules, libraries, or programs themselves may contribute to suspicious behaviour inside a container.

A crucial aspect of container protection is monitoring container processes and file system operation and identifying unusual behaviour. It will all identify irregular activities such as port scanning and reverse shells, or privilege escalations. A mixture of built-in identification and a simple behavioural learning method will be accessible that can classify suspicious behaviours focused on prevention.

If containerized systems are developed with microservice concepts in mind, because each program in a container has a small range of features, and the container is constructed with just the necessary packages and libraries, it is much simpler and more reliable to identify irregular processes and file system operation.

3.2.3 Host Security

If the host (e.g. Kubernetes worker node) on which containers run is running is compromised can lead to numerous security challenges. These include:

- **Privilege escalations to root** - The root access to the system may provide the attacker to modify the system in any way desired, it may also revoke permissions to the system thus preventing the user from any access to the system.

- **Changing of cluster-admin privileges** - The Cluster admin configuration to authentication modules may be changed to provide unrestricted permissions to unauthorized users.

- **Host resource damage or hijacking (e.g. crypto mining software)**. The attacker may launch bidirectional mount propagation in unprivileged containers to damage the host resources.

- **Stopping API Server or the Docker daemon**. These are the critical orchestration tools that are essential to ensure scalability and reliability within the containers.

3.3 Securing the Kubernetes System and Resources

Orchestration software like Kubernetes and their management platforms developed on top of it may be susceptible if not secured against attacks. This reveals possibly new attack surfaces for previously non-existent container implementations, which would thus be intended to be abused by hackers [9].

It's important to correctly configure the RBACs (Role-based access control) for device resources to secure Kubernetes and management systems themselves from attacks. Below are the areas for reviewing and configuring proper controls on access.

- **Protect the API Server** - To avoid unauthorized entry, customize RBAC for the API Server, or manually build firewall rules.

- **Restrict Kubelet Permission** - Set up RBAC for Kubelets and handle rotation of certificates to protect the Kubelet.

- **Authentication for all the external ports**. Check all publicly available ports and uninstall unused ports.

- **Mandatory authentication of external ports**. For non-authenticated services, restrict access to a whitelisted source.

- **Limit Console Access** - If correctly designed for user login with strong passwords or two-factor authentication, do not require console / proxy access.

4. CONCLUSION

The Kubernetes deployment network can be secure from attacks when paired with reliable host protection as mentioned before for locking down the worker nodes. However, surveillance software can also be used to control links to monitor networks and prevent unwanted attempts at connections and possible attacks. Real-time, policy-

based server, servers, network, and device resources tracking can identify all irregular activities and unwanted external connections.

REFERENCES

[1] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe and F. Khendek, "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, 2018, pp. 970-973.

[2] L. P. Dewi, A. Noertjahyana, H. N. Palit and K. Yedutun, "Server Scalability Using Kubernetes," 2019 4th Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), Bangkok, Thailand, 2019, pp. 1-4.

[3] C. Chang, S. Yang, E. Yeh, P. Lin and J. Jeng, "A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, 2017, pp. 1-6.

[4] D. Elliott, C. Otero, M. Ridley and X. Merino, "A Cloud-Agnostic Container Orchestrator for Improving Interoperability," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, 2018, pp. 958-961.

[5] A. Modak, S. D. Chaudhary, P. S. Paygude and S. R. Ldate, "Techniques to Secure Data on Cloud: Docker Swarm or Kubernetes," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, 2018, pp. 7-12.

[6] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," in IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014.

[7] D. Bringhenti, G. Marchetto, R. Sisto, F. Valenza and J. Yusupov, "Towards a fully automated and optimized network security functions orchestration," 2019 4th International Conference on Computing, Communications and Security (ICCCS), Rome, Italy, 2019, pp. 1-7.