Вестник УрФО № 2(44) / 2022, с. 63–69

DOI: 10.14529/secur220209

Ghadeer Darwesh, Jaafar Hammoud, Vorobeva A.A.

SECURITY IN KUBERNETES: BEST PRACTICES AND SECURITY ANALYSIS

Kubernetes emerged as docker containers' most popular orchestration, it is widely used for developing microservices and deploying applications. Because of advancements in containerization technology, information technology organizations use Kubernetes to manage their systems and report benefits in the deployment process. However, security concerns have been highlighted as challenges in Kubernetes deployment, The hackers can exploit the security vulnerabilities to cause damage to company assets. This work will shed the light on the Kubernetes orchestration platform and how attacks can be contacted against subevents manifest. we also demonstrate 10 security best practices in the Kubernetes cluster based on practitioners' reports, which we should follow to help protect our infrastructure.

Keywords: *Kubernetes, security, security policies, security practices, container security.*

Introduction

In recent years, microservices architecture has become more important and helped increase the software agility, containers emerged as the standard to deploy applications and microservices to the cloud. Nowadays this architecture is used by a big number of organizations to deliver their software such as Amazon, Netflix, Twitter, and other [1]. Kubernetes is an open-source software system used in microservices architecture such as cloud computing, the Internet of Things (IoT), and AI workflow. It has emerged as the most popular platform to manage the docker container life cycle and automate the management of computerized services.

From the viewpoint of cybersecurity, the Kubernetes system still has its own security challenges, and the users reported their concerns related to Kubernetes security. This work aims to explain potential exploits in the Kubernetes cluster and help users in securing their Kubernetes installation and deployment platform related to Kubernetes security best practices.

Section 2 states the background of this work and some important explanations related to it. Section 3 explains the threats and most recent security challenges in the Kubernetes system. In the last section, we present the Kubernetes security best practices for securing the Kubernetes deployment environment.

Backend

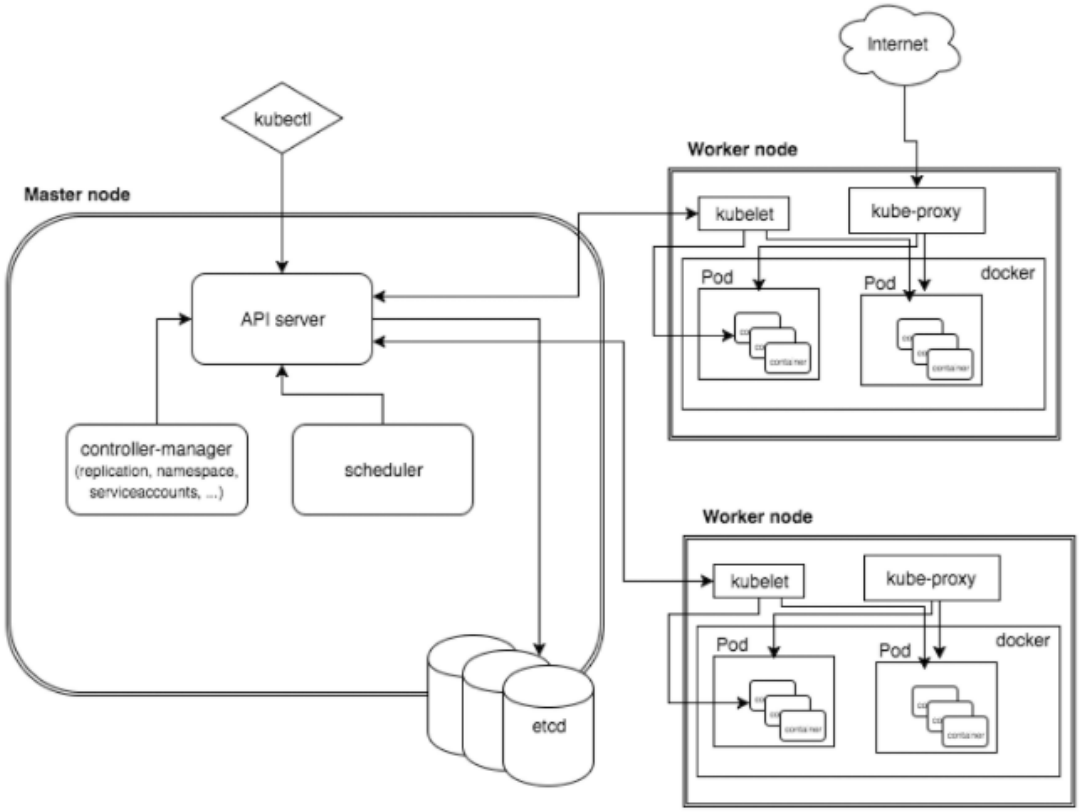
Kubernetes, at its fundamental level, maybe a framework for running and planning containerized applications over a cluster of machines. It disposes of most of the existing manual forms, which include the scaling, deploying, and managing of containerized applications [2]. Furthermore, based on utilization Kubernetes can scale the services up or down, ensuring we're only running what we would, like after we require it, wherever we need it. Like containers, Kubernetes permits us to oversee clusters, empowering the setup to be form controlled and replicated.

For a certain working scale, it gets to be fundamental to design our applications as a distributed system. Kubernetes is planned to supply the infrastructural layer for such desired systems, yielding clean applications to construct applications on top of a cluster [3]. More particularly, Kubernetes gives an interface to connect and manage this cluster such simply without needing to communicate individually with each machine.

Kubernetes follows the architecture of client-server architecture. It's conceivable to have a multi-master setup, but by default, there's a single master which acts as a controller and

point of contact [4]. The master server comprises different components counting a Kube-apiserver, an etcd, a Kube-controller-manager, a cloud-controller-manager, a kube-scheduler, and a DNS server for Kubernetes services. Node components incorporate kubelet and kube-proxy on top of Docker.

The Kubernetes master controls and



facilitates all nodes within the cluster with the assistance of three components that run on one or more master nodes within the cluster. Each Kubernetes master in our cluster runs these three processes:

1. Kube-apiserver: the single point of administration for the whole cluster. The API server implements a RESTful interface to communicate with tools and libraries. The kubectl command interacts directly with the API server.

2. Kube-controller-manager: controls the state of the cluster by managing all different kinds of controllers.

3. Kube-scheduler: schedules and plans the workloads across the nodes available in the cluster.

The entities state within the system is presented by Kubernetes Objects at any given point in time. Kubernetes Objects too act as an extra layer of abstraction over the container interface. We will presently specifically connect with instances of Kubernetes Objects rather than connection with containers. The fundamental Kubernetes objects are as follows:

Pod is the littlest deployable unit on the nodes. It's a bunch of containers that must run together. Very regularly, but not fundamentally, the pod contains one container.

Service is used to characterize a logical set of Pods and related policies utilized to access them.

Volume is basically a directory accessible to all containers running in a Pod. Namespaces are virtual clusters supported by the physical cluster.

Kubernetes security

Kubernetes workloads are vulnerable to several types of security threats, including:

• Compromise of the control plane—basic components like the API Server and etcd are not enough secured by default. The attacker gaining access to a Kubernetes master node can get control of the entire cluster.

• Compromise of pods and nodes—an attacker can get access to a physical host running Kubernetes pods, or to the individual

pods, enabling exfiltration of data within the pods.

• Compromise of network connections—pods and containers may be able to freely connect to each other and could be exposed to the public Internet. Any such open connections are an entry point for the attacker.

• Compromise of containers—containers can become vulnerable, due to a vulnerability or misconfiguration, or a backdoor in the container image. Containers with excess privileges can allow attackers access to the physical host.

Kubernetes security is imperative throughout the container lifecycle due to the dispersed, dynamic nature of a Kubernetes cluster. Distinctive security approaches are required for each of the three stages of an application lifecycle: build, deploy, and runtime. Kubernetes gives intrinsic security advantages. For case, application containers are regularly not fixed or updated — instead, container images are replaced totally with new versions. This enables strict version control and grants fast rollbacks in case a vulnerability is revealed in new code.

Despite the detailed benefits, recent studies show that security is one of the essential concerns for practitioners[5]. The study result from the StackRox [9] suggests that more than 44% of organizations delay their deployment for security concerns. The result moreover illustrates that 94% of the organizations have confronted at least one security occurrence within the final 12 months, among which 69% of security issues are misconfiguration-related [6]. The Cloud Computing (CNCF) study [6] result appears that 32% of practitioners among 1,324 overview members consider that security is their essential challenge in Kubernetes. Recent occurrences of security breaches give the authenticity of the professionals' concern.

According to the security risks characterized in NIST SP-800-190, most of the security solutions are appropriate for image pretest and also for operating system and network layer discovery within the host [7]. For protection against network attacks, we can introduce IPS, IDS, and web application firewalls within the network layer. In the operating system layer, we can install an antivirus program and host IDS for assurance against malware attacks. As there are numerous services on a single host in the runtime container environment, inner activities or anomalous behavior for each container cannot be followed by previous security solutions; only external communication attacks may be identified. This limits the ability to recognize containers that are subject to hacking attacks.

Related work

To collect our Internet artifacts. We use the Google search engine with 3 search strings: "Kubernetes security" 'Kubernetes security best practices, 'Kubernetes security policies".

[8] explored the availability of Kubernetes using a set of tests, and detailed that service outages can happen frequently. Shah and Dubaria [9] compared management features of Kubernetes, Docker Swarm, and Google Cloud Platform, and watched Kubernetes provide features, such as monitoring, deployment, and easy scalability. Takahashi etal Burns et al. [10] described the advancement of container management systems at Google and described how two internal systems called Omega and Borg evolved into Kubernetes.

Container security: SANS12 presented several tools for container security. For example, AppArmor13 was introduced as a policy-based Linux kernel security module that helps system administrators restrict process capabilities, such as file read/write permissions or network access, through their own security profiles[7].

[11] proposed a Kubernetes portable load balancer and detailed improved portability without sacrificing performance. Song et al. [12] constructed an auto-scaling system for API gateways using Kubernetes. The authors [12] reported that their constructed system could improve the utilization of system resources and ensure high availability. Muralidharan et al. [12] presented a Kubernetes-based system to manage and monitor Internet of Things (IoT) applications for smart cities.

reviewer [13]introduced a case study on Kubernetes and talked about how key concepts of Kubernetes can be utilized to streamline the scaling of containers. Medel et al. [14] collected real data from Kubernetes and used it to apply formal modeling to characterize resource management in Kubernetes. Chang et al. [15] presented a monitoring platform that uses Kubernetes to dynamically provision cloud resources.

KUBERNETES SECURITY PRACTICES

In this section we describe 10 Kubernetes security best practices reported by practitioners that can take enterprise security to the next level:

Authentication and Authorization Role-Based Access Control (RBAC) [9]

Authentication in Kubernetes alludes to the

verification of API requests through authentication plugins [16]. Authorization refers to the assessment of each authenticated API request against all policies to permit or deny the request [16].

One of these available plugins is the role-based access control (RBAC) plugin which allows the customer to control who can access the Kubernetes API and what permissions they have [17].

Practitioners have detailed a set of tasks to actualize the practice of authentication and authorization:

• In case we upgraded from a very old Kubernetes release and had not enabled RBAC earlier, RBAC settings should be checked to make sure they are enabled.

• Anonymous access to the Kubernetes server ought to be disabled. By default, Kubernetes permits anonymous get to the Kubernetes API server. [16] For case, a malicious user can figure out the default configuration of an insecure admission, gain access to the admission controller, and execute malicious commands.

• We should moreover manage the authorization approaches and utilize them properly. We use RBAC to limit groups and clients to just the activities and tasks they may need.

• Admission controller is a tool that intercepts requests to the Kubernetes API after the request is authorized, and before a volume is made persistent. Admission controllers have to be enabled and default authorization modes have to be disabled.

• We also should continuously follow the principle of least privilege to guarantee that users and Kubernetes service accounts have the minimal set of privileges required and make sure to not provide clusterwide authorizations, and don't deliver anybody cluster admin privileges unless completely necessary.

Private Kubernetes API Endpoint

Kubernetes cluster admins and operators can configure the Kubernetes API endpoint of a cluster as part of a public or private subnet. In the private cluster, the API server (endpoint) inside the control plane has a private IP address that makes the master blocked off from the public internet. In expansion to private worker nodes, we should make sure to configure the Kubernetes API endpoint as a private endpoint.

Kubernetes-specific Security Policies

Network policies Containerized applications generally make utilize cluster networks. We watch active network activity and compare it to the traffic permitted by Kubernetes network policy, to recognize how our application interacts and identify anomalous communications. By default, all Kubernetes pods have the ability to communicate with other pods. Practitioners suggest approaches to reduce network exposure, restrict traffic between pods, and restrict API server access to secure the network. In case firewalls are not set and network policies are not defined, at that point anybody may attack the API server from any IP address.

Pod-specific policies: It's recommended to apply security context to pods and containers. Pod policies define how the workloads should run in the Kubernetes cluster. Implementing workloads without defining a secure context for pods can make the Kubernetes cluster vulnerable, where a container can run with root privilege and write permission into the root file system. Practitioners also recommend that containers inside a pod must run as a non-root user with enabling Linux security modules and read-only permission.

Audit Logging Monitor Network Traffic to Limit Communications

It is recommended to enable audit logging and save audit logs to a secure repository to analyze the event of a compromise. Kubernetes moreover gives cluster-based logging to record container activity into a central logging subsystem. The standard output and error output of each container in a Kubernetes cluster can be ingested using an agent like Fluentd running on each node into tools like Elasticsearch and seen with Kibana. And at last, monitor pods, containers, services, and other components of our cluster using tools such as Prometheus and Grafana for monitoring, visibility, and following our cluster's state.

Namespace separation

A 'namespace' in Kubernetes is a logically isolated virtual cluster inside the physical cluster. [16] The creation of namespaces enables resources to be isolated between namespaces. Practitioners suggest that each group in a company should have a separate namespace for better manageability and running its production and development and environments.

Isolate Kubernetes Nodes

In addition to OS security, it is recommended that nodes are on a private network and not accessible from outside. A gateway may be

configured to get other services outside the cluster network, if required network ports to access nodes should be controlled via access lists. It is additionally recommended to restrict Secure Shell (SSH) access to the nodes.

Keep Kubernetes Version Up to Date

We should always update our cluster and run the most recent version of Kubernetes. This helps us to keep updated with all new security patches and Kubernetes updates. Upgrading Kubernetes can be a complex process case we're using a hosted Kubernetes provider.

Encrypt and restrict access to etcd

Etcd stores the state of the cluster and its secrets, so it is a sensitive asset and an attractive target for attackers. If unauthorized users could access the etcd, they can take over the whole cluster. Read access is additionally dangerous since malicious users can utilize it to elevate privileges.

The security practice is to encrypt the etcd storage. Encryption is important for securing etcd, and it's not enabled by default. We can enable it via kube-apiserver process, bypassing the argument "–encryption-provider-config" within the configuration, we need to select a provider to perform encryption and define our secret keys. [17]. Practitioners recommend restricting access to 'etcd', to only be available from the API servers, and isolated behind a firewall.

Limit CPU and memory quota

If a malicious user begins a denial of service (DOS) attack within a pod inside the Kubernetes cluster at that point, due to a high volume of requests, Kube-scheduler will create a new pod and an instance of the container will begin inside the new pod. This process proceeds until it consumes all available CPU resources and memory leaving all the applications in starvation. Practitioners recommend defining the number of resources by defining the maximum amount of memory for a namespace or a pod, the number of CPU shares for an application to consume, and the maximum number of instances for a container.

Enable SSL/TLS support

Practitioners suggest enabling TLS and SSL certificates for Kubernetes components. Enabling transport layer security (TLS) or secure sockets layer (SSL) protocol to ensure secure and encrypted communication between Kubernetes components.

Conclusion and future work

Over the years, containerization has steadily appeared its potential edges in the market. Developers use container innovation and serverless computing to solve various real-world challenges, such as VM's auto-scaling, performance loss issues, optimizing fetched, stack adjusting, and numerous others. Kubernetes is getting to be an attractive choice for keeping up containers for practitioners and organizations. Securing the Kubernetes system requires more consideration as default configurations of Kubernetes are frequently unreliable and insecure. Our paper appears that secure and effective utilization of Kubernetes requires the implementation of security practices applicable for numerous components inside the Kubernetes establishments: pods, containers, 'etcd' database, etc. This work helps practitioners to secure their Kubernetes installations system. Further, our current findings can lay the basis for conducting research in Kubernetes security.

References

1. Sultan S., Ahmad I., Dimitriou T. Container security: Issues, challenges, and the road ahead // IEEE Access. 2019. T. 7. C. 52976–52996.

2. Hightower K., Burns B., Beda J. Kubernetes: Up and running: Dive into the future of Infrastructure. Beijing, China: O'reilly, 2017.

3. Mondal S.K. и др. Kubernetes in it administration and Serverless Computing: An empirical study and research challenges // The Journal of Supercomputing. 2021. T. 78. № 2. C. 2937–2987.

4. Zhu H., Gehrmann C. Kub-SEC, an automatic Kubernetes Cluster APPARMOR Profile Generation Engine // 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS). 2022. C. 129–137.

5. Shamim S.I. Mitigating security attacks in Kubernetes manifests for security best practices violation // Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2021. C. 1689–1690.

6. 94% of organizations have suffered insider data breaches, egress research reveals [Электронный

ресурс]. URL: https://www.businesswire.com/news/home/20210713005123/en/94-Of-Organizations-Have-Suffered-Insider-Data-Breaches-Egress-Research-Reveals (дата обращения: 05.06.2022).

7. Tien C.W. и др. Kubanomaly: Anomaly Detection for the Docker orchestration platform with neural network approaches // Engineering Reports. 2019. Т. 1. № 5.

8. Abdollahi Vayghan L. и др. Deploying microservice based applications with Kubernetes: Experiments and Lessons Learned // 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). 2018. С. 970–973.

9. Shah J., Dubaria D. Building modern clouds: Using Docker, Kubernetes & Google Cloud Platform // 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). 2019. С. 184–189.

10. Burns B. и др. Borg, Omega, and kubernetes // Queue. 2016. Т. 14. № 1. С. 70–93.

11. Aida K., Tanjo T., Sun J. A portable load balancer for kubernetes cluster // Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region. 2018. С. 222–231.

12. Song M., Zhang C., Haihong E. An auto scaling system for API gateway based on Kubernetes // 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). 2018. С. 109–112.

13. Brewer E.A. Kubernetes and the path to cloud native // Proceedings of the Sixth ACM Symposium on Cloud Computing. 2015. С. 167–167.

14. Medel V. и др. Modelling Performance & Resource Management in Kubernetes // Proceedings of the 9th International Conference on Utility and Cloud Computing. 2016. С. 257–262.

15. Chang C.-C. и др. A Kubernetes-based monitoring platform for Dynamic Cloud Resource Provisioning // GLOBECOM 2017 - 2017 IEEE Global Communications Conference. 2017. С. 1–6.

16. Kubernetes Documentation [Электронный ресурс]. URL: https://kubernetes.io/docs/home/ (дата обращения: 05.06.2022).

17. Islam Shamim M.S., Ahamed Bhuiyan F., Rahman A. Xi commandments of Kubernetes Security: A systematization of knowledge related to Kubernetes Security Practices // 2020 IEEE Secure Development (SecDev). 2020. С. 58–64.

Ghadeer Darwesh, Ph.D. Student ITMO. Kronverksky pr., 49, St. Petersburg, Russia,197101. E-mail: ghadeerdarwesh32@gmail.com

Jaafar Hammoud, Ph.D. Student ITMO. St. Petersburg, Kronverksky pr., 49, St. Petersburg, Russia, 197101. E-mail: hammoudgj@gmail.com

VOROBEVA Alisa Andreevna, Associate professor ITMO. Kronverksky pr., 49, St. Petersburg, Russia, 197101. E-mail: alice_w@mail.ru

УДК 004.42 + 004.49 + 004.056

Гадир Дарвеш, Джафар Хаммуд, Воробьева А.А.

БЕЗОПАСНОСТЬ В ПЛАТФОРМЕ KUBERNETES: РЕКОМЕНДАЦИИ И АНАЛИЗ БЕЗОПАСНОСТИ

Платформа Kubernetes появилась как самая популярная оркестровка контейнеров Docker и широко используется для разработки микросервисов и развертывания приложений. Благодаря усовершенствованию технологии контейнеризации, организации в сфере информационных технологий применяют платформу Kubernetes для управления своими системами и создания отчетов о положительных эффектах в процессе развертывания. Однако специалисты обращают внимание на возможные проблемы обеспечения безопасности при развертывании с использованием платформы Kubernetes. Хакеры могут воспользоваться уязвимыми местами системы безопасности, чтобы нанести вред активам компании. Данная работа прольет свет на платформу оркестровки Kubernetes и на то, как проявляются атаки на подсобытия и как справляться с ними. Мы также приводим основанные на отчетах практикующих специалистов 10 рекомендаций по безопасности для кластера платформы Kubernetes, которые следует соблюдать для обеспечения защиты инфраструктуры.

Ключевые слова: *платформа Kubernetes, безопасность, правила разграничения доступа, рекомендации по безопасности, безопасность контейнера.*

Гадир Дарвеш, аспирант Университета ИТМО. Кронверкский пр., 49, Санкт-Петербург, Россия, 197101. Электронная почта: ghadeerdarwesh32@gmail.com

Джафар Хаммуд, аспирант Университета ИТМО. Санкт-Петербург, Кронверкский пр., 49, Санкт-Петербург, Россия, 197101. Электронная почта: hammoudgj@gmail.com

ВОРОБЬЕВА Алиса Андреевна, доцент Университета ИТМО. Кронверкский пр., 49, Санкт-Петербург, Россия, 197101. Электронная почта: alice_w@mail.ru