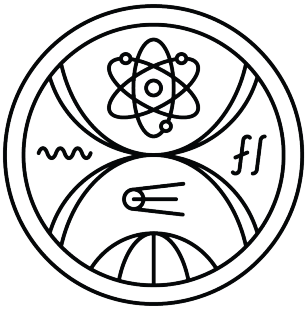
COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS

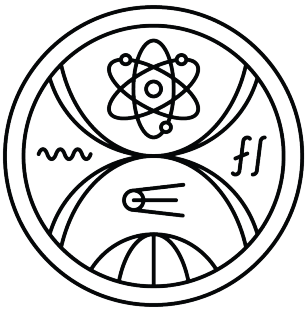


KUBERNETES SECURITY ASSESSMENT

Master thesis

2024 Bc. Pavel Semenov

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS PHYSICS AND INFORMATICS



KUBERNETES SECURITY ASSESSMENT

Master thesis

Study program:	Applied informatics
Branch of study:	Applied informatics
Department:	Department of Applied Informatics
Supervisor:	prof. RNDr. Richard Ostertág, PhD.
Consultant:	Mgr. Ľubomír Firment

Bratislava, 2024 Bc. Pavel Semenov

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname: Bc. Pavel Semenov
Study programme: Applied Computer Science (Single degree study, master II. deg., full time form)
Field of Study: Computer Science
Type of Thesis: Diploma Thesis
Language of Thesis: English
Secondary language: Slovak

Title: Kubernetes security assessment

Annotation: Kubernetes has been gaining popularity rapidly in recent years as more and more enterprise solutions are subjected to cloud transformation and more companies are looking for the ways to increase development efficiency and reduce development costs. This brings new concerns from clients and stakeholders about the security of Kubernetes and its exposure to cyber-attacks.

Aim:

This thesis studies, compares and evaluates the state-of-the-art tools designed to discover vulnerabilities concerning the cluster configuration, running pods or cluster itself. Assessment is carried out in both local cluster setup predisposed with multiple vulnerabilities and real-world enterprise cloud infrastructure. Based on the assessment results we intend either to improve one of the existing tools or develop a Kubernetes security framework of our own, which will be able to provide better results in addressing the cluster security.

Literature:

V. B. Mahajan and S. B. Mane, "Detection, Analysis and Countermeasures for Container based Misconfiguration using Docker and Kubernetes", 2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS), 2022, pp. 1-6, doi: 10.1109/IC3SIS54991.2022.9885293. https://ieeexplore.ieee.org/document/9885293

D. B. Bose, A. Rahman and S. I. Shamim, "'Under-reported' Security Defects in Kubernetes Manifests", 2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS), 2021, pp. 9-12, doi: 10.1109/EnCyCriS52570.2021.00009. https://ieeexplore.ieee.org/document/9476056

Castillo Rivas, D.A., Guamán, D. (2021). "Performance and Security Evaluation in Microservices Architecture Using Open Source Containers". In: Botto-Tobar, M., Montes León, S., Camacho, O., Chávez, D., Torres-Carrión, P., Zambrano Vizuete, M. (eds) Applied Technologies. ICAT 2020. Communications in Computer and Information Science, vol 1388. Springer, Cham. https://doi.org/10.1007/978-3-030-71503-8_37

Clinton Cao, Agathe Blaise, Sicco Verwer, and Filippo Rebecchi (2022). "Learning State Machines to Monitor and Detect Anomalies on a Kubernetes Cluster". In Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22). Association for Computing Machinery, New York, NY, USA, Article 117, 1–9. https://doi.org/10.1145/3538969.3543810

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Computing Machinery, New York, NY, USA, Article 117, 1–9. https://doi.org/10.1145/3538969.3543810

Vedúci: RNDr. Richard Ostertág, PhD.
Konzultant: Mgr. Ľubomír Firment
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 07.12.2022

Dátum schválenia: 07.12.2022 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

I hereby declare that I have written this thesis by myself, only with help of referenced literature, under the careful supervision of my thesis advisor.

.....

Bratislava, 2024 Bc. Pavel Semenov

Acknowledgement

First, I would like to express my gratitude to Mgr. Ľubomír Firment for his guidance during the whole thesis and invaluable expertise in Kubernetes that made this thesis possible. I'd also like to thank my supervisor prof. RNDr. Richard Ostertág, PhD. for his insightful feedback.

Abstract

Kubernetes has been gaining popularity rapidly in recent years as more and more enterprise solutions are subjected to cloud transformation and more companies are looking for the ways to increase development efficiency and reduce development costs. This brings new concerns from clients and stakeholders about the security of Kubernetes and its exposure to cyber-attacks. This thesis studies, compares and evaluates the state-of-the-art tools designed to discover vulnerabilities concerning the cluster configuration, running pods or cluster itself. Assessment is carried out in both local cluster setup predisposed with multiple vulnerabilities and real-world enterprise cloud infrastructure. Based on the assessment results we intend either to improve one of the existing tools or develop a Kubernetes security framework of our own, which will be able to provide better results in addressing the cluster security.

Keywords: kubernetes, security, test, cloud

Abstrakt

Kubernetes v posledných rokoch rýchlo získava na popularite, pretože čoraz viac spoločností hľadá spôsoby, ako zvýšiť efektivitu vývoja a znížiť náklady na vývoj. Táto zvýšená popularita so sebou prináša väčšie vystavenie kybernetickým útokom a zvýšené obavy zainteresovaných strán o bezpečnosť Kubernetes. Cieľom práce je porovnať a zhodnotiť moderné nástroje určené na odhaľovanie zraniteľností týkajúcich sa konfigurácie klastra, bežiacich podov alebo aj samotného klastra. Posúdenie bude prebiehať na lokálnom klastri s prednasadenými viacerými zraniteľnosťami, ako aj v reálnej podnikovej cloudovej infraštruktúre. Na základe výsledkov hodnotenia máme v úmysle buď vylepšiť niektorý z existujúcich nástrojov, alebo vyvinúť vlastný bezpečnostný rámec pre Kubernetes, ktorý bude schopný poskytnúť lepšie výsledky pri riešení klastrovej bezpečnosti.

Kľúčové slová: kubernetes, bezpečnosť, testovanie, cloud

Contents

List of Figures

List of Tables

Terminology

Add terms and abbreviations as we encounter them in our text.

Terms

- **CI/CD pipeline**
  CI/CD pipeline is a set of automatic tasks to be executed upon specific action resulting in ether failure on some of the pipeline stages or successful application deployment on the target environment. The aforementioned action might be a push into the source code repository or a manual pipeline run request. CI/CD pipeline usually includes build, test and deploy stages.
- **Cloud**
  Term "Cloud" is ususally used to describe an array of (remote, on-premise or hybrid) servers that operate as a single ecosystem used for various hosting services.

Abbreviations

- **K8s** - Kubernetes.
- **OS** - Operating System.
- **VM** - Virtual Machine.
- **CI/CD** - Continuous Integration/Continuous Deployment.

Motivation

As the world's biggest corporations start grasping the power of the Cloud Computing, stakeholders are raising concerns regarding the security of the most popular and accessible Container Orchestration platform - Kubernetes. Opinion of the experts on this matter varies significantly and this paper aims to make a contribution to this dispute by determining how well can be Kubernetes cluster's security monitored.

We compare and evaluate the capabilities of the most popular security tools designed specifically for Kubernetes against official and unofficial Kubernetes security recommendations.

add results

Chapter 1

Introduction

1.1 Kubernetes

In this section, we will dive into the topic of Kubernetes. We will define containerization and compare it to the traditional means of application deployment. We will expand on the Kubernetes resources, their kinds and their role in the target application environment. Finally, we will go through the official Kubernetes security recommendations.

1.1.1 Definition

Kubernetes, also known as K8s, is an open source system for automating deployment, scaling, and management of containerized applications. [1]. IBM defines Kubernetes as an open source **container orchestration platform** for scheduling and automating the deployment, management and scaling of containerized applications. [3].

1.1.2 Containerization

When talking about the Kubernetes it is essential to define the term containerization. Containerization is the packaging of software code with just the operating system libraries and dependencies required to run the code to create a single lightweight executable—called a container—that runs consistently on any infrastructure. [2].

Although containers are built to be infrastructure-agnostic, there are still certain compatibility considerations to keep in mind. One significant factor is processor architecture. Containers built for a specific architecture family (e.g., arm64, amd64, or x86) are generally not cross-compatible with infrastructures based on different architectures. However, it is possible to build multi-architecture containers that support multiple processor architectures in a single image, enhancing the flexibility and portability of the containerized applications across diverse environments.

1.1.3 Containerization vs Virtualization

Let us discuss why containerization is rapidly growing in popularity nowadays and why do software arhitects tend to choose it over traditional virtualization solutions.

I would say the most important advantage of the containers is their resource efficiency. Containers only include the application code and its dependencies, which makes them very small compared to the Virtual Machines, which tend to be very bulky and grow in size as development progresses. Containers share the host operating system kernel, so they consume significantly less CPU, memory, and storage than virtual machines, which require a full OS for each instance. This lightweight nature allows more containers to run on a single host, maximizing resource utilization and reducing overhead. Better resource efficiency means also lower costs for the user.

Then, startup speeds are significantly lower for the containers as they do not need to initialize the whole OS boot sequence. This feature also enables the scaling capabilities for the containers, allowing applications to respond quickly to changes in demand.

Additionally, the containers are more consistent than virtual machines. Packed with the required dependencies, they behave in the same way across different environments. As they are isolated from the OS, containers are almost immune to the compatibility issues. This gives them a strong portability advantage. They provide an abstraction that makes it easier to move workloads across various platforms.

Lastly, containers also lead when it comes to automation and CI/CD pipelines. Containers can be easily versioned, updated, and rolled back, allowing for smooth integration into CI/CD pipelines. This streamlines deployment, testing, and rollback processes, leading to faster development cycles. VMs can also be updated and rolled back, but the process is usually slower and more complex.

These are some of the most significant advantages of containerization. For large-scale development, test and production environment containerization is an obvious choice over the virtualization. While costing less money and providing a lot more flexibility, they become essential for successful enterprise software development.

1.1.4 Kubernetes resources

Kubernetes resources are fundamental components that define various entities within a Kubernetes cluster. Resources are objects that represent the desired state and configuration of the infrastructure, applications, and services running on the cluster. Kubernetes provides a range of resources that enable developers and operators to define, manage, and scale containerized applications, network policies, storage requirements, and more. These resources are defined declaratively in YAML or JSON files, which makes infrastructure setup consistent and reproducible.

Among key Kuberntes resoucres are:

- Pods Pod is the atomic workload unit in the Kuberntes cluster. It encapsulates one or more containers that share the same network. It represents a single instance of a running application.
- Deployments Deployments are a higher level of abstraction for the Pods. They allow to define replica count and rollout/rollback strategy for the updates, which can be used to ensure availability for the application.
- Services Services provide a communication layer for the pods inside one cluster. Being an abstraction over the pods' network, they provide reliable access to the selected workloads, while serving as a Load Balancer.
- ConfigMaps and Secrets ConfigMaps and Secrets allow users to store data outside the workload. ConfigMaps are usually used to store non-sensitive information like environment and application configuration parameters. Secrets are a more secure resource designed for API keys, passwords and other sensitive data.
- PersistentVolumes and PersistentVolumeClaims These resources enable stateful applications to request and mount durable storage within a cluster, allowing data to persist independently of the Pod lifecycle.

Above are the most commonly used resources, which we also leverage in the practical part of the paper. Therefore, it is important that the reader understands the position and the purpose of each resource in the cluster infrastructure.

1.1.5 Kubernetes security recommendations

This section gathers the official security recommendations provided by the Kubernetes. They provide a list of concerns for each level of the cloud infrastructure. Cloud infrastructure can be viewed as a composition of four layers.

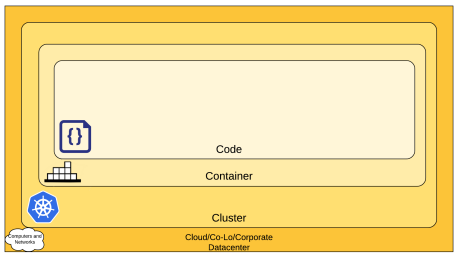


Figure 1.1: Four layers of the cloud infrastructure.

Each layer is built upon the previous one and its security depends on the security of the outer layers. It is, therefore, important to maintain high security standarts on base levels (Cloud, Cluster, Container).

1. **Cloud**

   Each cloud provider has its own security policies and guidelines. There are, however, some general infrastructure-level security best advice:

Area of Concern for Kubernetes Infrastructure	**Recommendation**
Network access to API Server (Control plane)	All access to the Kubernetes control plane is not allowed publicly on the internet and is controlled by network access control lists restricted to the set of IP addresses needed to administer the cluster.
Network access to Nodes (nodes)	Nodes should be configured to only accept connections (via network access control lists) from the control plane on the specified ports, and accept connections for services in Kubernetes of type NodePort and LoadBalancer. If possible, these nodes should not be exposed on the public internet entirely.
Kubernetes access to Cloud Provider API	Each cloud provider needs to grant a different set of permissions to the Kubernetes control plane and nodes. It is best to provide the cluster with cloud provider access that follows the principle of least privilege for the resources it needs to administer.
Access to etcd	Access to etcd (the datastore of Kubernetes) should be limited to the control plane only. Depending on your configuration, you should attempt to use etcd over TLS.
etcd Encryption	Wherever possible it's a good practice to encrypt all storage at rest, and since etcd holds the state of the entire cluster (including Secrets) its disk should especially be encrypted at rest.

2. **Cluster**

   There are two cluster security concerns that could be addressed: securing the configurable cluster components and securing the applications running in the cluster. Security of cluster components is described in **Cluster security** chapter.

   There are a few things to consider regarding the application security:

   - RBAC Authorization (Access to the Kubernetes API)
   - Authentication
   - Application secrets management (and encrypting them in etcd at rest)
   - Ensuring that pods meet defined Pod Security Standards
   - Quality of Service (and Cluster resource management)
   - Network Policies
   - TLS for Kubernetes Ingress

3. **Container**

   Securing containers is a vast topic, which deserves its own chapter. There are, nevertheless, a few general recommendation provided by the Kubernetes:

Area of Concern for Containers	**Recommendation**
Container Vulnerability Scanning and OS Dependency Security	As part of an image build step, you should scan your containers for known vulnerabilities.
Image Signing and Enforcement	Sign container images to maintain a system of trust for the content of your containers.
Disallow privileged users	When constructing containers, create users inside of the containers that have the least level of operating system privilege necessary in order to carry out the goal of the container.

4. **Code**

   When it comes to code, the developers have the most flexibility to design secure applications. There are a lot of issues to address, which may vary significantly from application to application depending on its purpose, architecture and framework base. Kubernetes documentation gives a handful of recommendations regarding this topic.

Area of Concern for Code	**Recommendation**
Access over TLS only	If your code needs to communicate by TCP, perform a TLS handshake with the client ahead of time. With the exception of a few cases, encrypt everything in transit. Going one step further, it's a good idea to encrypt network traffic between services. This can be done through a process known as mutual TLS authentication or mTLS which performs a two sided verification of communication between two certificate holding services.
Limiting port ranges of communication	This recommendation may be a bit self-explanatory, but wherever possible you should only expose the ports on your service that are absolutely essential for communication or metric gathering.
3rd Party Dependency Security	It is a good practice to regularly scan your application's third party libraries for known security vulnerabilities. Each programming language has a tool for performing this check automatically.
Static Code Analysis	Most languages provide a way for a snippet of code to be analyzed for any potentially unsafe coding practices. Whenever possible you should perform checks using automated tooling that can scan codebases for common security errors.
Dynamic probing attacks	There are a few automated tools that you can run against your service to try some of the well known service attacks. These include SQL injection, CSRF, and XSS. One of the most popular dynamic analysis tools is the OWASP Zed Attack proxy tool.

Bibliography

[1] The Linux Foundation. Kubernetes official web. https://kubernetes.io/, Accessed: 4.11.2024.

[2] Ian Smalley (IBM) Stephanie Susnjara (IBM). What is containerization? https://www.ibm.com/topics/containerization, Accessed: 4.11.2024.

[3] Ian Smalley (IBM) Stephanie Susnjara (IBM). What is kubernetes? https://www.ibm.com/topics/kubernetes, Accessed: 4.11.2024.