FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

# Hardening Kubernetes Clusters

Reducing Attack Surface in Kubernetes by means of Rootless Containers, Network Policies and Role Based Access Control

**Bachelor Thesis**

Submitted in partial fulfillment of the requirements for the degree of

**Bachelor of Science in Engineering**

to the University of Applied Sciences FH Campus Wien

Bachelor Degree Program: Computer Science and Digital Communications


**Author:**

Guntram Björn Klaus

**Student identification number:**

c2110475170

**Supervisor:**

BSc. MSc. Bernhard Taufner


**Date:**

dd.mm.yyyy

Declaration of authorship:

I declare that this Bachelor Thesis has been written by myself. I have not used any other than the listed sources, nor have I received any unauthorized help.

I hereby certify that I have not submitted this Bachelor Thesis in any form (to a reviewer for assessment) either in Austria or abroad.

Furthermore, I assure that the (printed and electronic) copies I have submitted are identical.

Date: Signature:

# Abstract

(E.g. “This thesis investigates...”)

# Kurzfassung

(Z.B. “Diese Arbeit untersucht...”)

# List of Abbreviations

| | |
|---|---|
| ARP | Address Resolution Protocol |
| GPRS | General Packet Radio Service |
| GSM | Global System for Mobile communication |
| WLAN | Wireless Local Area Network |

# Key Terms

GSM

Mobilfunk

Zugriffsverfahren

# Contents

# 1 Introduction

## 1.1 Background: Enterprises and Cloud

## 1.2 Research Objectives

## 1.3 Methodology

## 1.4 Structure

# 2 Concepts

## 2.1 Containervirtualization

### 2.1.1 Linux Kernel

### 2.1.2 Container Images

### 2.1.3 Container Runtimes

## 2.2 Kubernetes

### 2.2.1 Components

### 2.2.2 Cluster Architecture

### 2.2.3 Cluster Objects

# 3 Literature Review

Kubernetes is highly customizable and offers a range of configuration options which determine the security posture of individual applications and the cluster as a whole. The purpose of this literature review is to explore the security threat landscape of Kubernetes environments. Specifically, recurring concepts and common denominators across vulnerabilities shall be identified and discussed. For this, the database of Common Vulnerabilities and Exposures (CVE), the IEEE database, the ACM digital library and the official Kubernetes feed of CVEs are queried using keywords pertaining to Container and Kubernetes Security. The acquired papers, articles and CVE descriptions are skimmed through. The most relevant results are narrowed down and selected for closer inspection. It shall be noted that Kubernetes vulnerabilities do not only entail standard Kubernetes components, but also add-ons deployed on top of 'plain' Kubernetes. Such can be the Nginx Ingress-Controller, a Service-Mesh, CI/CD tools closely embedded into Kubernetes and more. Generally, this can be anything that extends the Kubernetes API through Custom Resource Definitions (CRDs).

It is evident that the reliance on Kubernetes comes with the need of a clear security initiative. According to RedHat's report on the state of Kubernetes security of 2022, more than ninety percent of polled organizations underwent at least one security incident in their Kubernetes environment, which, in a third of cases, lead to the loss of revenue or customers. The majority of these incidents were detections of misconfigurations. About a third of respondents reported major vulnerabilities and runtime security incidents in relation to containers and/or Kubernetes which required immediate remediation. In a more recent, similar report conducted by RedHat in 2023, two thirds of respondents had to delay or slow down application deployments because of security concerns. This is a significant increase compared to the 2022 survey, where just over half of participants experienced delays. Three of the most frequently mentioned advantages of containerization include quicker release cycles, quicker bug fixes, and increased flexibility to operate and manage applications. But if security is neglected, you can lose out on containerization's biggest benefit: agility. It becomes apparent that Kubernetes is not something that is installed once and then never looked at again. Rather, a container-based environement that leverages this orchestration technology requires attention for detail and constant, rigurous inspection, despite the great amount of abstraction provided and due to its highly customizable nature. CITE REDHAT 2022 2023

Container Escape Major vulnerabilities include those which fall under the category of a so called container-escape. Since a container is intended to be a runtime environment isolated from the underlying host, the concept of a container-escape relates to performing an exploit that breaks the confines of exactly this isolation, resulting in full or limited access to the underlying host machine and/or network. A study conducted by Reeves et al. at the end of 2021 investigates the susceptibility of different container runtime systems to escape-exploits by studying a batch of CVE reports. The study identifies three main causes for container escapes. First, mishandled file descriptors, if for example left accessible from within a container under /proc, enables malicious actors read and write access to the underlying host filesystem, as seen in CVE-2019-5736. In this reported vulnerability, a container is set up with a symlink from the container's entrypoint to proc/self/exe, which points back to its

runC binary, which instantiated the container process. In addition, the container carries a harmful file which is designed to overwrite the file descriptors of any excecuting process that loads it. If an unkowing person executes a binary within the container, which has been manipulated to symlink to /proc/self/exe, the harmful file is able to overwrite the runC binary. The next time another, unrelated container is spawned, it is done by the compromised runC binary. Secondly, missing access control to runtime components could enable adversaries to gain access to UNIX sockets on the host, as reported in CVE-2020-15257. Here, it was possible to connect to the containerd socket, thus enabling actors to issue API commands to freely create new containers on the host, unconstrained by Apparmor, seccomp, or Linux capabilities. Thirdly, under 'adversary-controlled host execution' problems of similar fashion to mishandled file descriptors are mentioned. In this case however, vulnerability exposure starts with host binaries being executed in the container context, which makes it a target for manipulation. In CVE-2019-101(44–47), the shared library "libc.so.6" is altered in such a way that it mounts the host filesystem when loaded. The new shell loads "libc.so" when the administrator runs "rkt-enter", which is the /bin/bash command by default, to create a new shell in the container. This sets off malicious code embedded in "libc.so", which uses the mknod syscall to construct a block device of the host root filesystem inside the container. As a result, the adversary is able to read and write to the host filesystem.

CVE-2022-0811, which is barely discussed in papers due to its young nature, reports a sophisticated container escape possibility of the CRI-O container engine.

the foundation of such attacks is being able to either freely instantiate containers or freely move and operate from inside a container.

The most notable ones, which have a severity score above 8, according to the Common Vulnerability Scoring System (CVSS), are CVE-2022-0811 and such and such because they have been shown to provide full root access once successfully exploited.

We see that vulnerabilities vary tremendously in their appearance. CVE so and so targets a tool, CVE so and so targets the actual underlying kernel of a k8s node. However, all these things can be prevented or alleviated by applying the concept of least privilege to containers, access to Kubernetes and the Kubernetes network. For example, if a malicious actor cannot freely create files in any desired directory of a compromised container, it significantly reduces his ability to exploit a given vulnerability. Not being able to create that file in the first place, due to missing write permissions, would be a major obstacle for performing a container escape. It is not always possible to completely avoid a vulnerability. This is due to software bugs and unidentified weaknesses in the code that have passed through the testing and review process.

## 3.1 State of the Art

## 3.2 CVE Numbers

## 3.3 Incident Reports

# 4 Hardening Measures

## 4.1 Securing Containers

### 4.1.1 Linux File Permissions

### 4.1.2 Root versus Rootless

### 4.1.3 Rootless Policy Enforcement

## 4.2 Securing the Network

Default pod-to-pod network settings, as an example, allow open communication to quickly get a cluster up and running, at the expense of security hardening. Network segmentation.....

### 4.2.1 Network Policies

### 4.2.2 Service Mesh

## 4.3 Authentication and Authorization

### 4.3.1 Role Based Access Control

### 4.3.2 Kubeconfig Keys and Certificates

### 4.3.3 Service Accounts

# 5 Discussion

## 5.1 Implications for businesses

## 5.2 Tradeoffs and Difficulties

## 5.3 Complexity

# 6 Conclusion

# 7 Outlook / Future work

# Bibliography

# List of Figures

# List of Tables

# Appendix

(Hier können Schaltpläne, Programme usw. eingefügt werden.)