Kubernetes inter-pod container isolation

Aarni Halinen

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 1.6.2023

Thesis supervisor:

Prof. Mario Di Francesco

Thesis advisors:

M.Sc. (Tech.) José Luis Martin Navarro

M.Sc. (Tech.) Jacopo Bufalino

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Aarni Halinen

Title: Kubernetes inter-pod container isolation

Date: 1.6.2023 Language: English Number of pages: 5+10

Department of Computer Science

Professorship: Computer Science

Supervisor: Prof. Mario Di Francesco

Advisors: M.Sc. (Tech.) José Luis Martin Navarro, M.Sc. (Tech.) Jacopo Bufalino

Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained. Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained.
Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained. Your abstract in English. Try to keep the abstract short; approximately 100 words should be enough. The abstract explains your research topic, the methods you have used, and the results you obtained.

Keywords: Kubernetes, Container, Docker, Security

Preface

I want to thank Professor Pirjo Professori and my instructor Olli Ohjaaja for their good and poor guidance.

Otaniemi, 16.1.2015

Eddie E. A. Engineer

Contents

Symbols and abbreviations

Symbols

$\uparrow$	electron spin direction up
$\downarrow$	electron spin direction down

Operators

$\nabla \times \mathbf{A}$	curl of vectorin $\mathbf{A}$

Abbreviations

K8s	Kubernetes
STRIDE	an object-oriented analog circuit simulator and design tool

1 Introduction

Tämän tekstin lähteenä oleva tiedosto on opinnäytteen pohja, jota voi käyttää kandidaatintyössä, diplomityössä ja lisensiaatintyössä. Tekstin lähteenä oleva tiedosto on kirjoitettu LaTeX-tiedoston rakenteen opiskelemista ajatellen. Tiedoston kommentit sisältävät tietoa, joka on hyödyllistä opinnäytettä kirjoitettaessa.

Johdanto selvittää samat asiat kuin tiivistelmä, mutta laveammin. Johdannossa kerrotaan yleensä seuraavat asiat

- Tutkimuksen taustaa ja tutkimusaiheen yleisluonteinen esittely
- Tutkimuksen tavoitteet
- Pääkysymys ja osaongelmat
- Tutkimuksen rajaus ja keskeiset käsitteet.

Lyhyiden opinnäytteiden johdannot ovat yleensä liian pitkiä, joten johdannon paisuttamista on vältettävä. Diplomityöhön sopii johdanto, joka on 2–4 sivua. Kandidaatintyön johdannon on oltava diplomityön johdantoa lyhyempi. Sopivasti tiivistetty johdanto ei kaipaa alaotsikoita.

2 Background

2.1 Principle of least priviledge, Zero trust...

2.2 Microservices?

2.3 Containerization and Docker

2.3.1 Docker components

2.3.2 Linux control groups and namespaces

2.3.3 Linux capabilities, priviledged containers, Container breakout

[3]

1. Priviledged container
2. CAP_SYS_ADMIN, mounting /proc and chroot
3. CAP_SYS_PTRACE, shellcode injection to running program, nc 172.17.0.1 on port running shell
4. Mounted docker socket, creating priviledged containers

2.4 Kubernetes system components, control plane

2.4.1 apiserver

2.4.2 etcd

2.4.3 scheduler

2.4.4 controller-manager

2.5 Kubernetes resources

2.5.1 Namespaces

2.5.2 Pods

2.5.3 Services

2.5.4 Admission control

2.6 Kubernetes network model

Integral part of Kubernetes cluster is how nodes and resources are networked together. Specifically, the networking model needs to address four different type of networking problems: i) intra-Pod (ie. container-to-container within same Pod) communication, ii) inter-Pod communication between Pods, iii) Service-to-Pod communication and iv) communication from external sources to Services [1]. The model also requires that each Pod is IP addressable and can communicate with other Pods without network

address translation (NAT), even when Pods are scheduled on different hosts. [10]. All agents on a host should also be able to communicate with Pods on the same host. The implementation of this model is not part of Kubernetes, but is handed to special plugins that implement Container Network Interface (CNI) specification.

2.6.1 Container Network Interface

The Container Network Interface (CNI) [5] is a networking specification, which has become de facto industry standard for container networking. It is backed by Cloud Nativce Computing Foundation (CNCF) [10]. CNI was first developed for the container runtime `rkt`, but it is supported by all container runtimes and there is a large number of implementations to choose from [8]. Most of the container orchestrators have adopted the specification as their networking solution. The biggest outlier is Docker Swarm, which instead implements `libnetwork` [6].

The specification has five distinct definitions: i) a format for network configuration, ii) a execution protocol between the container runtimes and the plugin binary, iii) a procedure for the runtime to interpret the configuration and execute the plugins, iv) a procedure for deletegating functionality between the plugins and v) data types for plugins to return their results to the runtime [5]. The configuration is defined as a JSON file and it includes a list of plugins and their configuration. The file is consumed at plugin execution time by the runtime, and passed to the plugins. The execution protocol defines a set of operations (ADD, DEL, CHECK) for adding and removing containers from the network, while also defining a set of OS environment variables that are used as parameters by the plugins. When the runtime mutates a container network, it results in a series of ADD, DELETE or CHECK executions. These are then executed in same order as defined in the `plugins` list, or reversed order for DELETE executions. Each plugin then returns either `Success` or `Error` JSON object. The execution of a series of operations ends when it encounters the first Error response, or when all the operations have been performed.

2.6.2 Network policies

Since Kubernetes does not provide networking between the Pods, it has no capabilities to enforce network isolation between workloads. Thus, another key feature for CNI plugins is enforcing network traffic rules. Kubernetes provides a common resource called `NetworkPolicy` for CNI plugins to consume. The NetworkPolicy specification consists of a `podSelector` that specifies pods that are subject to the policy and `policyTypes` to specify Ingress and Egress rules for the traffic [2] to the target Pod. Each rule includes `to` or `from` field for selecting Pod, Namespace or IP address block in CIDR notation on the other side of the connection, and `ports` field for explicitly specifying which ports and protocols are part of the rule. The policies are additive; when multiple rules are defined for a Pod, the traffic is restricted to what is allowed by the union of the policies. Many CNI plugins also introduce Custom Resource Definitions for their own, more granular, network policy rules.

2.6.3 Cilium and other CNI plugins

TODO: CNI plugins, daemons and binary [11].

While all CNI plugins meet the requirements listed above, they may differ in architecture significantly. The plugins can be classified based on which OSI model network layer they operate on, which Linux kernel features they use for packet filtering and which encapsulation and routing model they support for inter-host and intra-host communication between Pods.

In this thesis, we focus on three different CNI plugins: Cilium, Calico and Multus.

Cilium [4] is one of the most advanced and powerful CNI plugins for Kubernetes. It works by creating virtual ethernet device for each Pod and sets one side of the link into Pod's network namespace [7]. Cilium then attaches extended Berkeley Packet Filter (eBPF) programs to ingress traffic control (`tc`) hooks of these virutal ethernet devices for intercepting all incoming packets from the Pod. The packets are intercepted and processed before the network stack and thus `iptables`, reducing latency 20%-30% and even doubling the throughput of packets in some scenarios [2].

Cilium provides Custom Resource Definition `CiliumNetworkPolicy` that supports policies in layers 3-7 instead of standard L3/L4. With `CiliumClusterwideNetworkPolicy`, network rules can applied to every namespace in the cluster, or even to nodes when using `nodeSelector`.

2.6.4 Extended Berkeley Packet Filter

Berkeley Packet Filter (BPF, or nowadays often cBPF) was originally developed in early 1990s as a high-performance tool for user-space packet captures [9]. BPF works by deploying the filtering part of the application, `packet filter`, in the kernel-space as an agent. The `packet filter` is provided with a program (often denoted as BPF program) consisting of BPF instructions, which works as a set of rules for selecting which packets are of interest in the user-space application and should be copied from kernel-space to user-space. The instuctions are executed in a register-based pseudo machine. Since network monitors are often interested only in subset of network traffic, this limits the number of expensive copy operations across the kernel/user-space protection boundary only to packets that are of interest in the user-space application. A notable usecase for BPF is *libpcap* library, which is used by network monitoring tool called `tcpdump`.

Later in the 2010s the Linux community realized that BPF and it's ability to instrument the kernel could benefit other areas than packet filtering as well [12]. This reworked version of BPF was first merged in to Linux kernel in 2014 and is publicly called extended Berkeley Packet Filter (eBPF) to distinguish it from the original cBPF. The kernel development community continues to call the newer version BPF, but instead of the original acronym consider it a name of a technology. Similarly to the kernel community, the term BPF always refers to the eBPF in this thesis.

1. eXpress Data Path
2. Traffic control

3 Research material and methods

Tässä osassa kuvataan käytetty tutkimusaineisto ja tutkimuksen metodologiset valinnat, sekä kerrotaan tutkimuksen toteutustapa ja käytetyt menetelmät.

4 Evaluation

Tässä osassa esitetään tulokset ja vastataan tutkielman alussa esitettyihin tutkimuskysymyksiin. Tieteellisen kirjoitelman arvo mitataan tässä osassa esitettyjen tulosten perusteella.

Tutkimustuloksien merkitystä on aina syytä arvioida ja tarkastella kriittisesti. Joskus tarkastelu voi olla tässä osassa, mutta se voidaan myös jättää viimeiseen osaan, jolloin viimeisen osan nimeksi tulee »Tarkastelu». Tutkimustulosten merkitystä voi arvioida myös »Johtopäätökset»-otsikon alla viimeisessä osassa.

Tässä osassa on syytä myös arvioida tutkimustulosten luotettavuutta. Jos tutkimustulosten merkitystä arvioidaan »Tarkastelu»-osassa, voi luotettavuuden arviointi olla myös siellä.

5 Conclusion

Opinnäytteen tekijä vastaa siitä, että opinnäyte on tässä dokumentissa ja opinnäytteen tekemistä käsittelevillä luennoilla sekä harjoituksissa annettujen ohjeiden mukainen muotoseikoiltaan, rakenteeltaan ja ulkoasultaan.

References

[1] The Kubernetes Authors. *Cluster Networking*. 2022. URL: https://kubernetes.io/docs/concepts/cluster-administration/networking/ (visited on 03/09/2023).

[2] Gerald Budigiri et al. "Network policies in kubernetes: Performance evaluation and security analysis". In: *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*. IEEE. 2021, pp. 407–412.

[3] Thanh Bui. "Analysis of docker security". In: *arXiv preprint arXiv:1501.02967* (2015).

[4] Cilium. *Cilium*. 2023. URL: https://cilium.io/ (visited on 03/14/2023).

[5] CoreOS. *CNI—The Container Network Interface*. 2023. URL: https://github.com/containernetworking/cni/blob/v1.1.2/SPEC.md (visited on 03/10/2023).

[6] Docker. *libnetwork*. 2023. URL: https://github.com/moby/moby/tree/master/libnetwork (visited on 03/10/2023).

[7] The Kubernetes Networking Guide. *Cilium*. 2023. URL: https://www.tkng.io/cni/cilium/ (visited on 03/14/2023).

[8] Michael Hausenblas. *Container Networking*. O'Reilly Media, Incorporated, 2018.

[9] Steven McCanne and Van Jacobson. "The BSD Packet Filter: A New Architecture for User-level Packet Capture." In: *USENIX winter*. Vol. 46. 1993.

[10] Shixiong Qi, Sameer G Kulkarni, and KK Ramakrishnan. "Assessing container network interface plugins: Functionality, performance, and scalability". In: *IEEE Transactions on Network and Service Management* 18.1 (2020), pp. 656–671.

[11] Shixiong Qi, Sameer G Kulkarni, and KK Ramakrishnan. "Understanding container network interface plugins: design considerations and performance". In: *2020 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE. 2020, pp. 1–6.

[12] Marcos AM Vieira et al. "Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications". In: *ACM Computing Surveys (CSUR)* 53.1 (2020), pp. 1–36.

A Esimerkki liitteestä

Liitteet eivät ole opinnäytteen kannalta välttämättömiä ja opinnäytteen tekijän on kirjoittamaan ryhtyessään hyvä ajatella pärjäävänsä ilman liitteitä. Kokemattomat kirjoittajat, jotka ovat huolissaan tekstiosan pituudesta, paisuttavat turhan helposti liitteitä pitääkseen tekstiosan pituuden annetuissa rajoissa. Tällä tavalla ei synny hyvää opinnäytettä.