

kdigger

Kubernetes focused container assessment and context discovery tool for penetration testing

Mahé Tardy (@mtardy_)

About me

Quarkslab

- **Mahé Tardy**
- Security R&D Engineer @ **Quarkslab**
- Doing research on **Kubernetes**
- **@mtardy_**
- **mtardy@quarkslab.com**



CVEs of March for container escapes

Linux

- **CVE-2022-0847** a.k.a. DirtyPipe. Vulnerability allows for overwrite of files that should be read-only.
- **CVE-2022-0492**. Vulnerability in cgroup handling can allow for container breakout depending on isolation layers in place.

CRI-O

- **CVE-2022-0811**. Vulnerability in setting sysctls in k8s/OpenShift manifests allows for container breakout.

Containerd

- **CVE-2022-23648**. Vulnerability in volume mounting allows for arbitrary file read from the underlying host, leading to likely indirect container breakout.

Source: <https://www.container-security.site/> by @raesene





We don't need that!

Linux

- CVE-2022-0810. A privilege escalation vulnerability. DirtyPipe. Vulnerability allows for overwrite of /proc/sys/fs/binfmt_misc/uapi/ directory which should be restricted by SELinux.
- CVE-2022-0811. A privilege escalation vulnerability. A bug in kernel handling can allow for container breakout if SELinux or AppArmor security protection layers in place.

CRI-O

- CVE-2022-0811. A privilege escalation vulnerability. A bug in kernel handling sysctls in k8s/OpenShift manifests can allow for container breakout.

Containerd

- CVE-2022-0811. A privilege escalation vulnerability. A bug in kernel mounting allows for arbitrary file creation on the host system via Containerd leading to likely information disclosure and breakout.

Source: <https://www.container-security.net/> by @raesene





Kubernetes security
today is more about
configuration than
vulnerabilities.



Why another tool?

Participated in the last 4 KubeCon Cloud-Native Security Day CTFs.

Learned the habits of many experts in the fields.

Loved **amicontained** by Jessie Frazelle.

Decided to automate a security checklist from inside a Kubernetes Pod!





What can kdigger do for you?

Kubernetes focused container assessment and context discovery tool for penetration testing

Like **amicontained** you can:

- Try to guess your container runtime.
- See your capabilities.
- Scan for namespace activation and configuration.
- Scan for the allowed syscalls.

And more **basic** stuff: check mounts, uid, processes, devices, status flag, and API versions.

But from a **Kubernetes** perspective:

- Retrieve service account token.
- Scan token permissions.
- List interesting environment variables.
- Retrieve all available services in a cluster.
- Retrieve leaked information by cgroups v1.
- Retrieve the specifications of the node.
- Scan the admission controller chain!
- Retrieve all the CRDs



What can kdigger do for you?

Kubernetes focused container assessment and context discovery tool for penetration testing

Like **amicontained** you can:

- Try to guess your container runtime.
- See your capabilities.
- Scan for namespace activation and configuration.
- Scan for the allowed syscalls.

Bla bla bla

And more basic stuff: check mounts, uid, processes, devices, status flag, and API versions.

Bla bla bla

But from a **Kubernetes** perspective:

- Retrieve service account token.
- Scan token permissions.
- List interesting environment variables.
- Retrieve all available services in a cluster.
- Retrieve leaked information by endpoints v1.
- Retrieve the specifications of the node.
- Scan the admission controller chain!
- Retrieve all the CRDs

Bla bla bla

Bla bla bla



Disclaimer on usage



Some checks rely on implementation details or fragile features:

- PID namespace or container runtime check from **amicontained**.
- CoreDNS wildcard feature for services listing has been removed in v1.9.0.

<https://github.com/coredns/coredns/issues/4984>

Needs to be updated and extended with new checks

Straightforward to extend, look at the contributing guide in the README if you have interesting checks to add.

Demonstration!

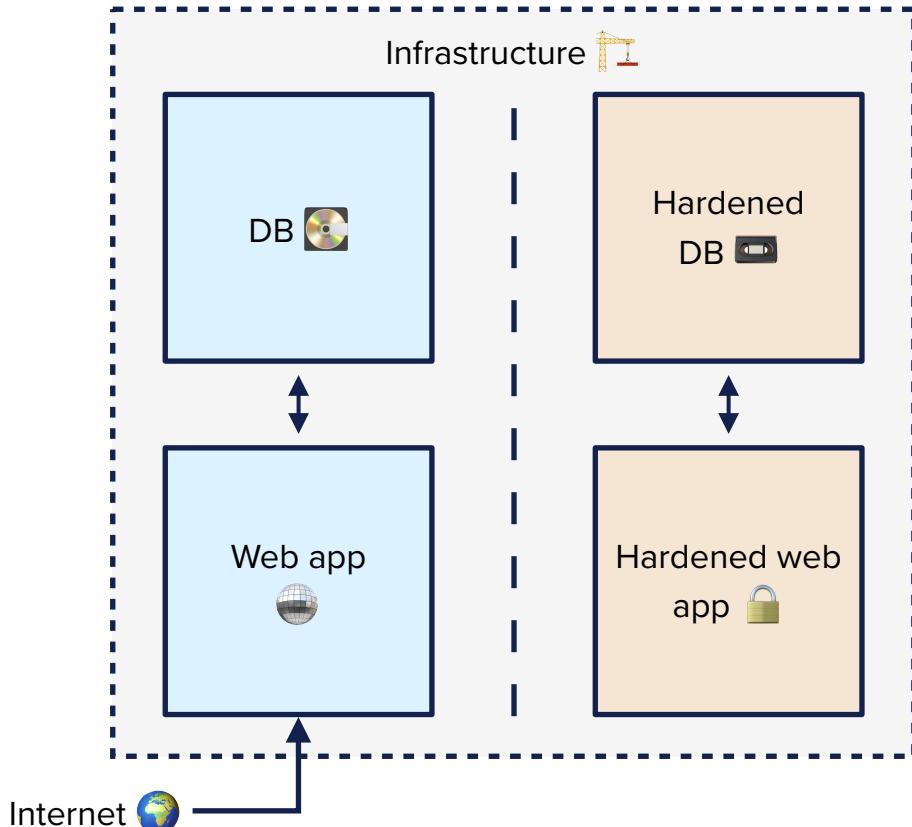


Simplified infrastructure

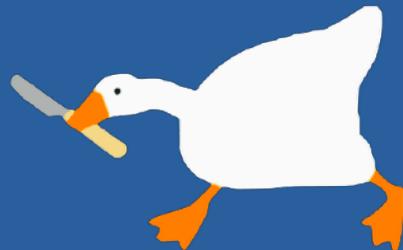
This setup was inspired by a mission and highlights the problems of multi tenancy in clusters.

Context:

- Web app exposed to internet.
- Hardened web app, only exposed on the internal network.



Honk time!

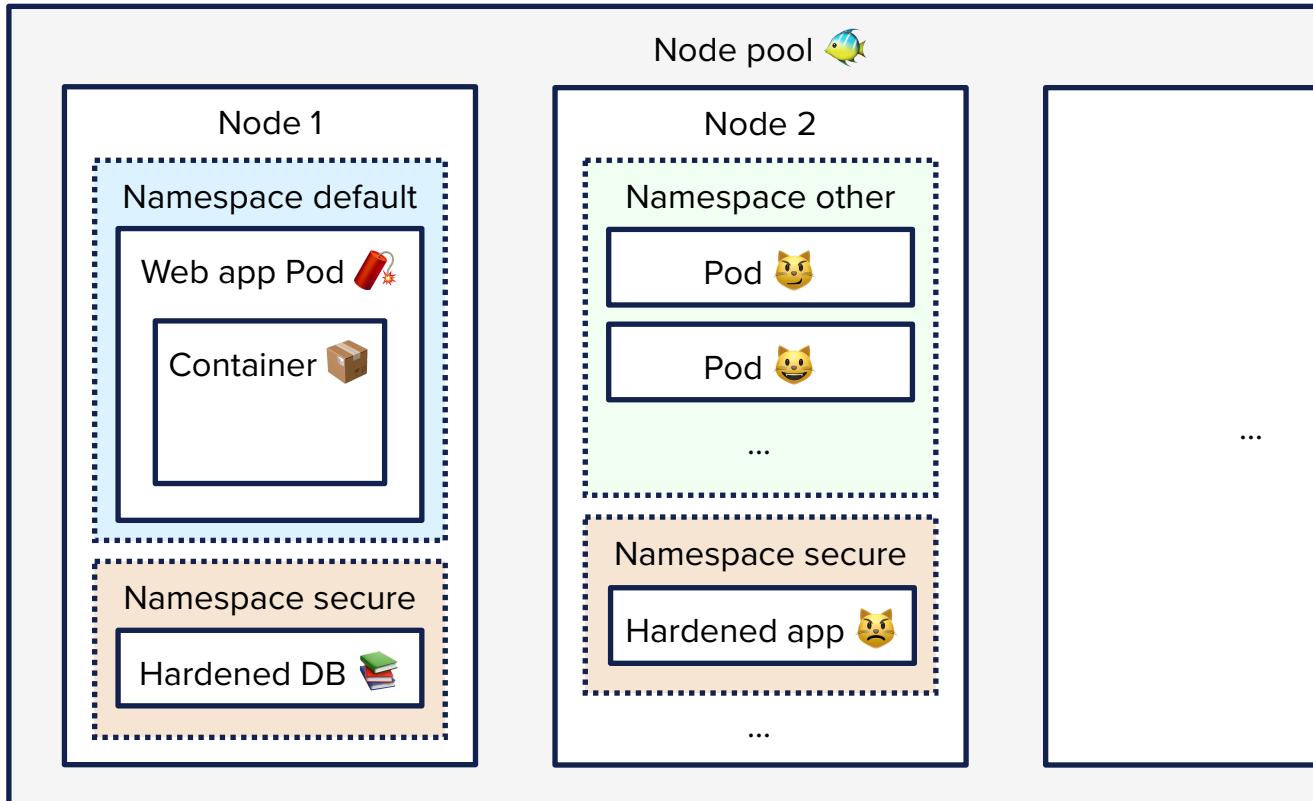
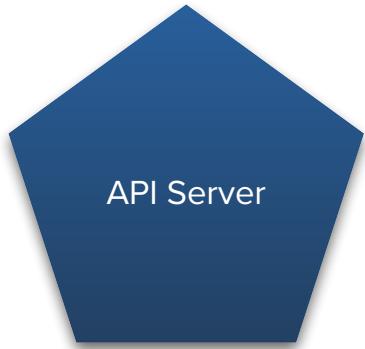


Quarkslab

What exactly happened?

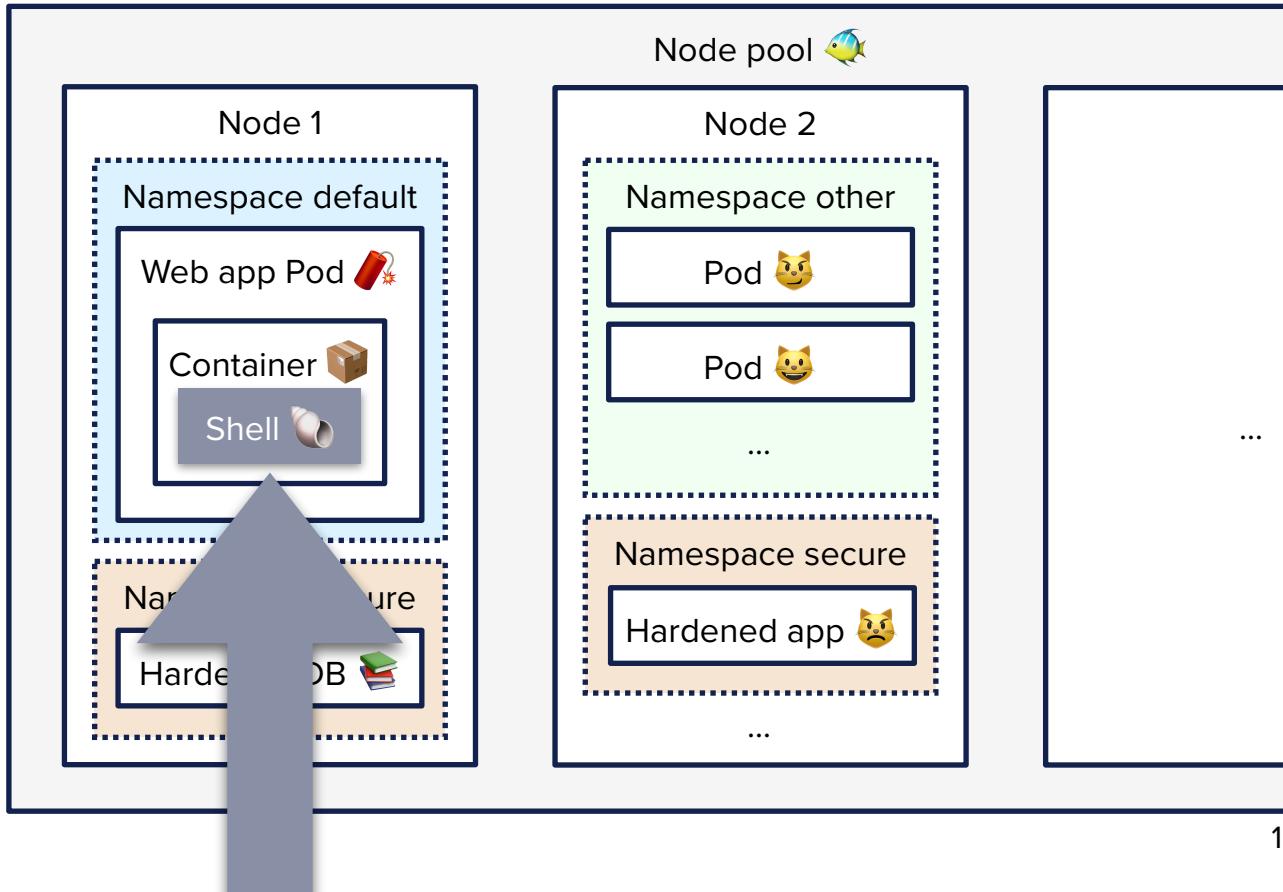
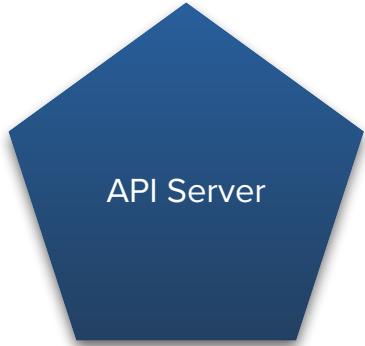


What happened? - Three main issues



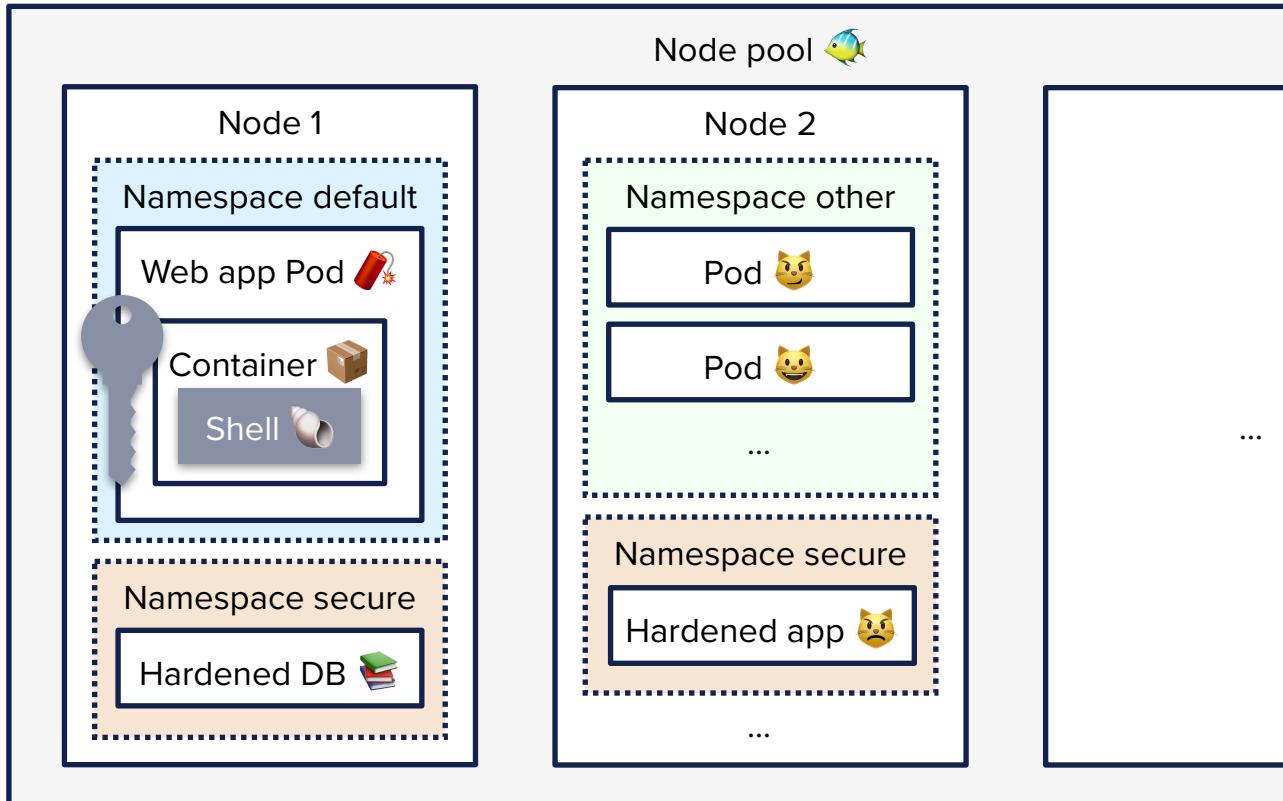
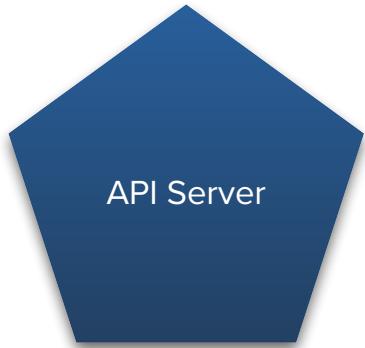


What happened? - First issue: vulnerability in software



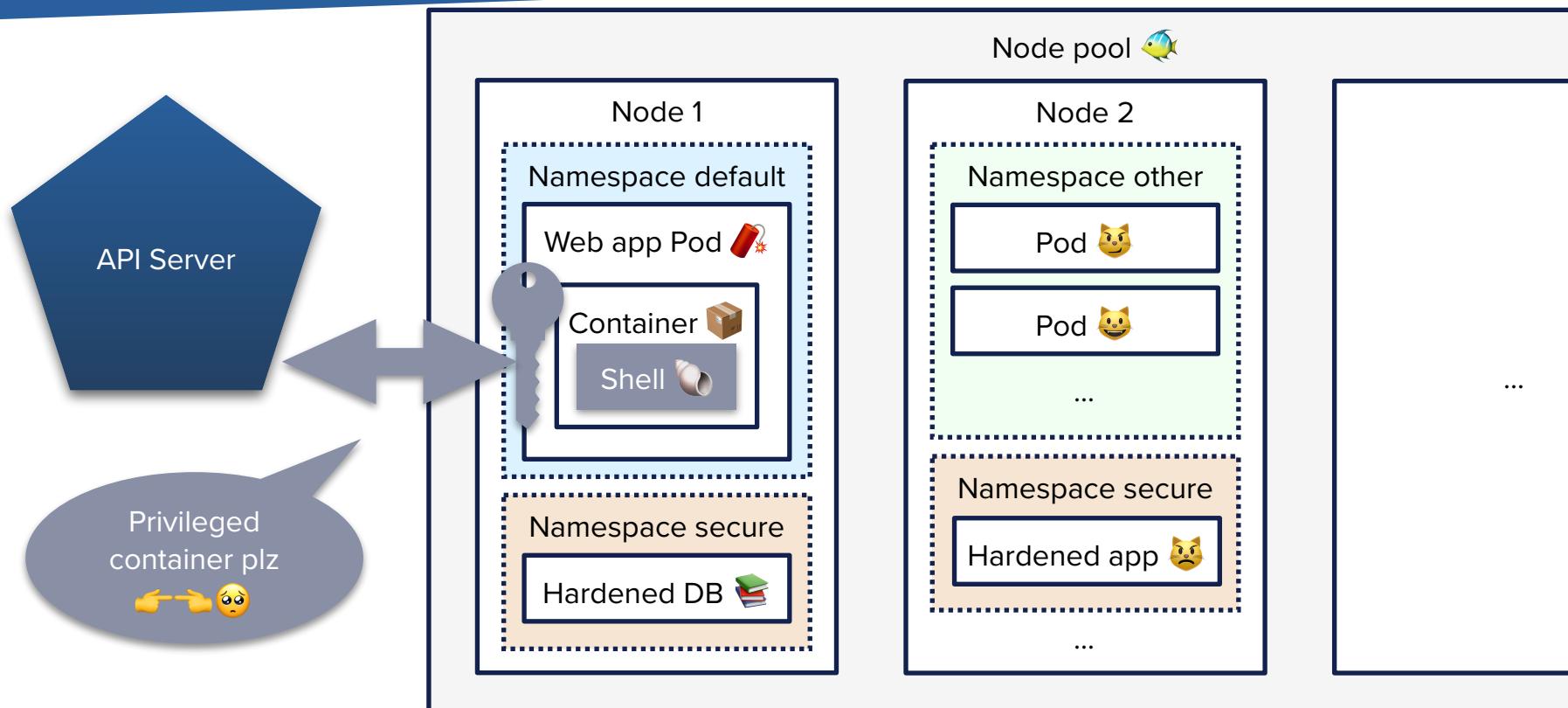


What happened? - Second issue: credentials in container



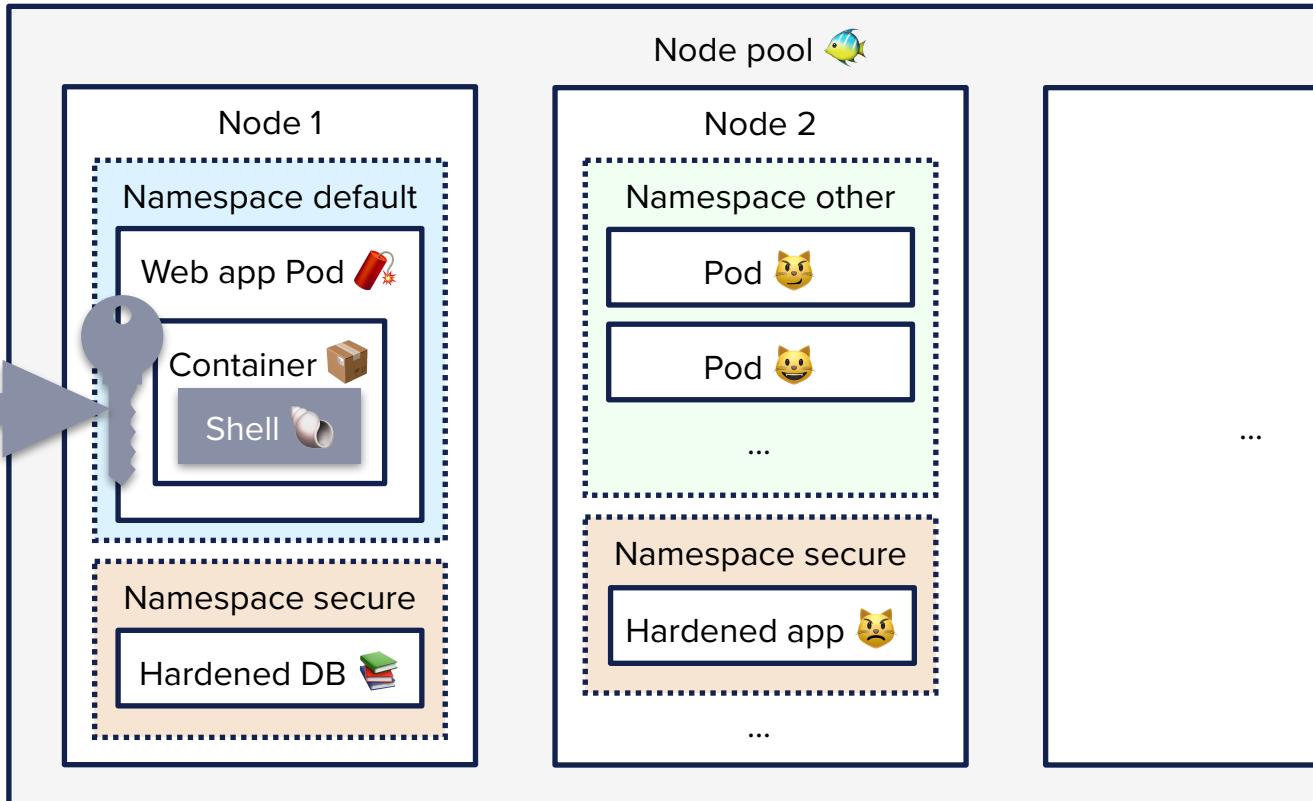
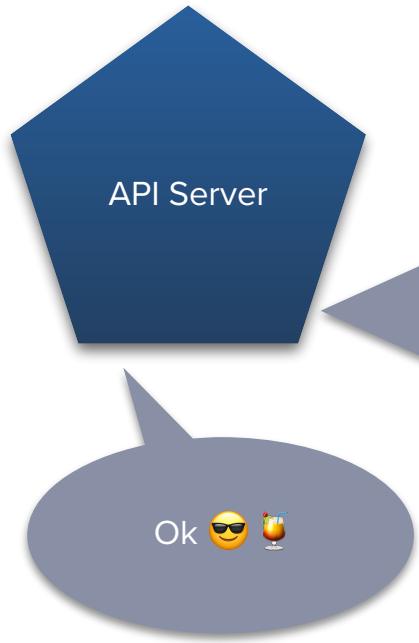


What happened? - Third issue: no admission control



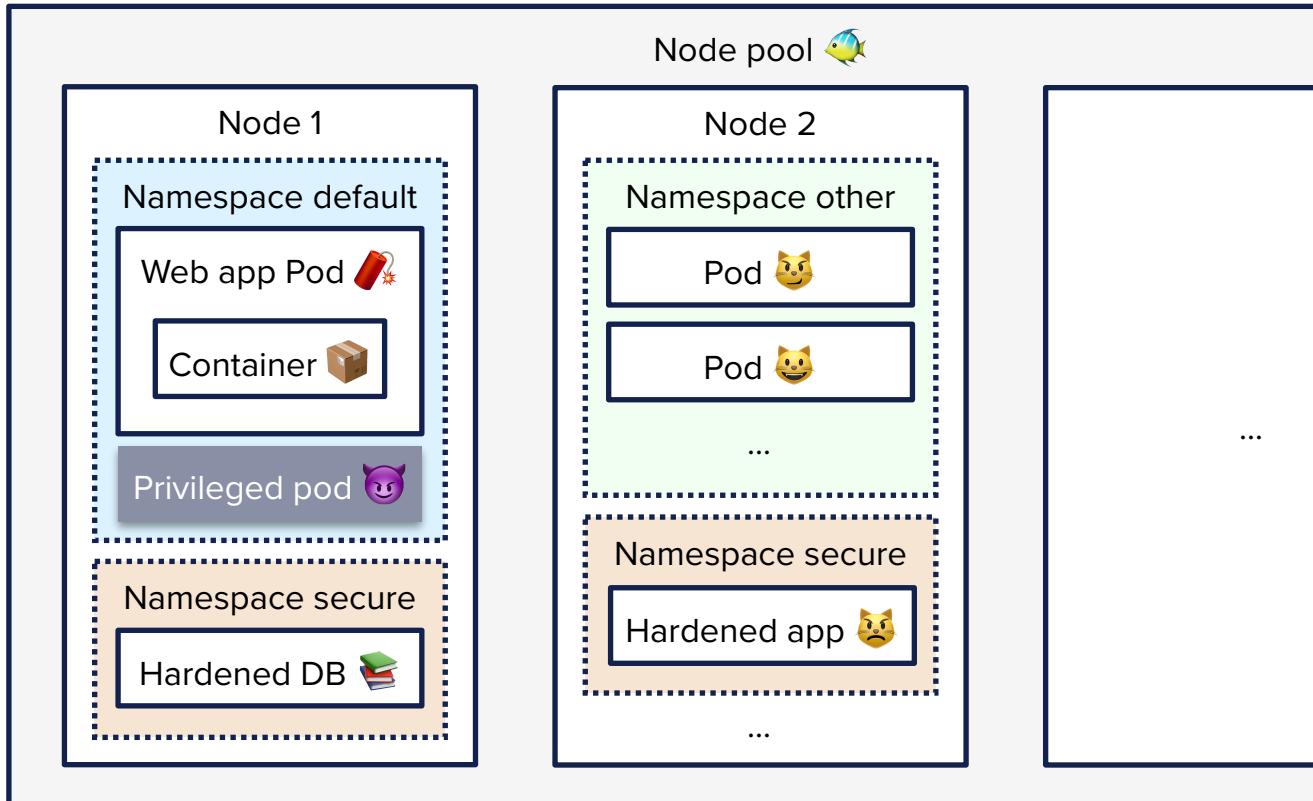
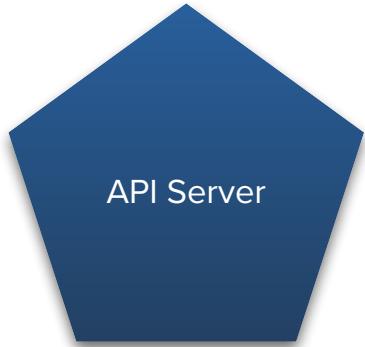


What happened? - Third issue: no admission control



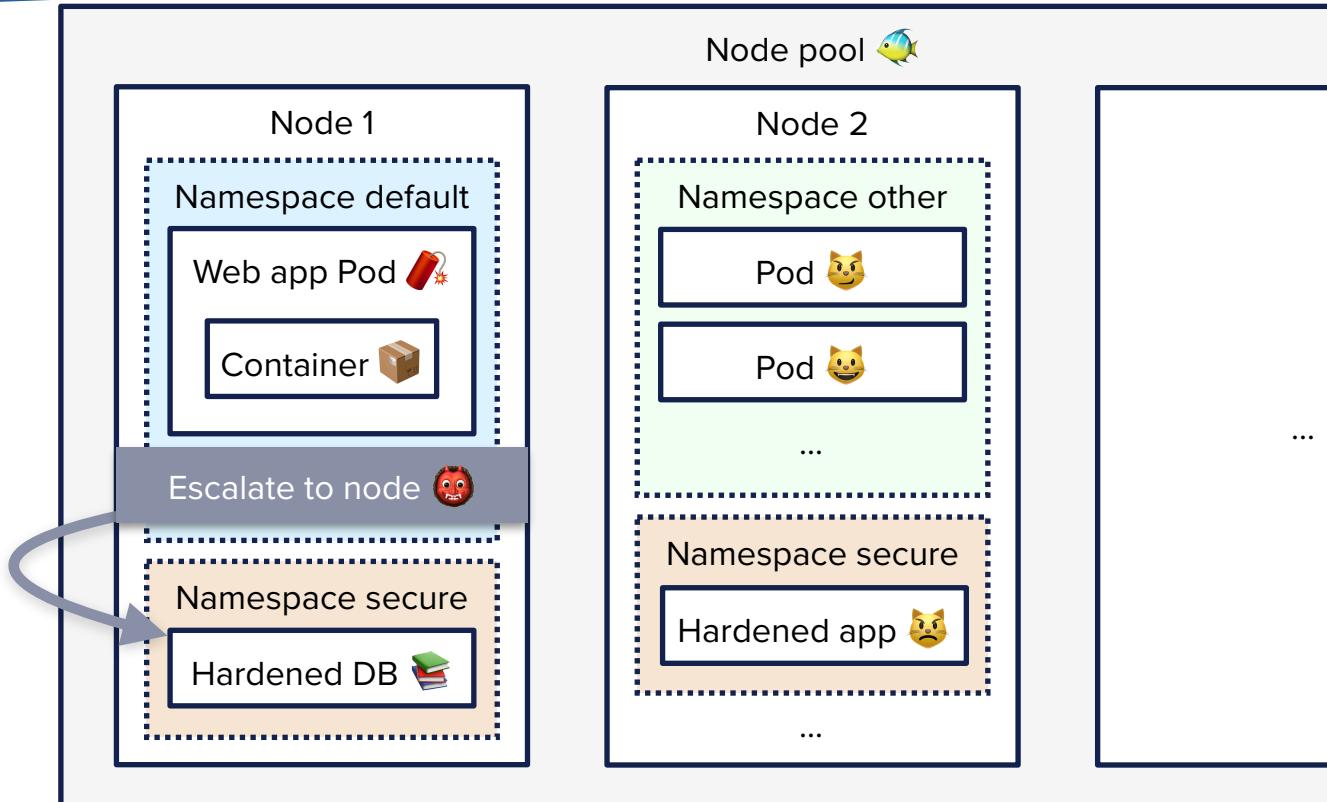
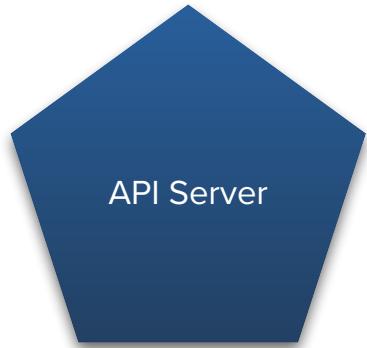


What happened? - Third issue: no admission control



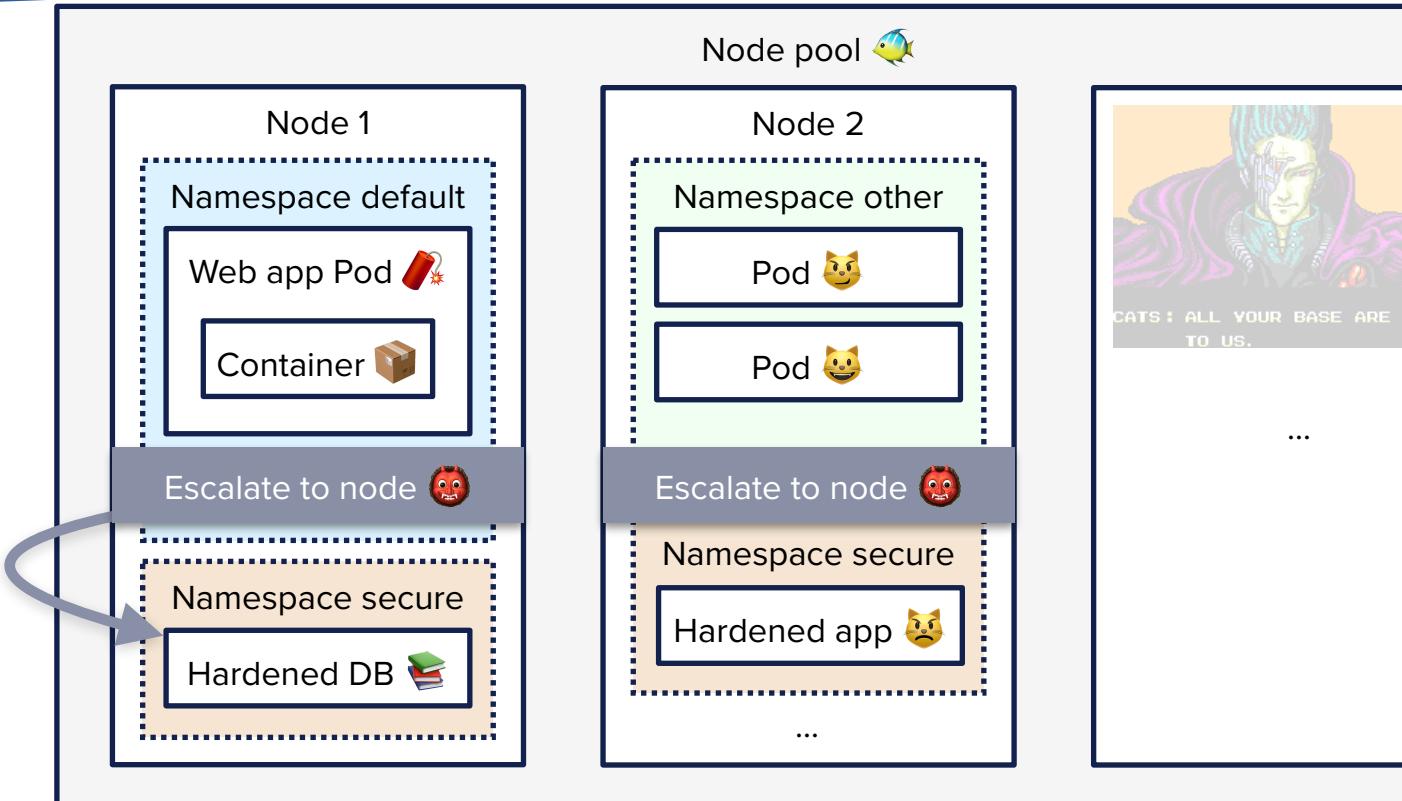
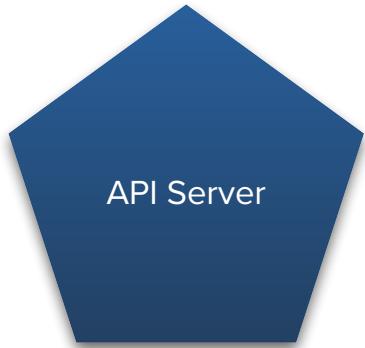


What happened? - Third issue: no admission control





What happened? - Third issue: no admission control



How to fix these issues?



First issue: vulnerabilities in software

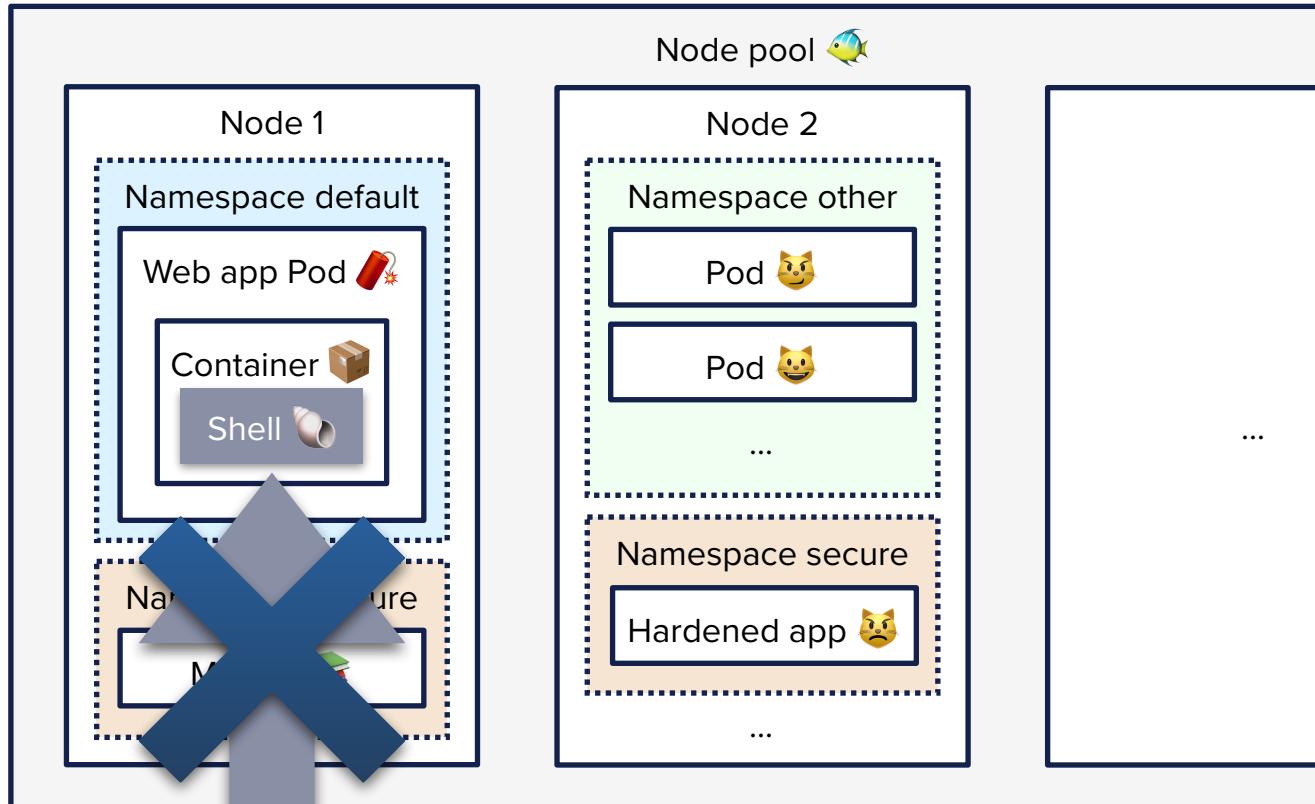
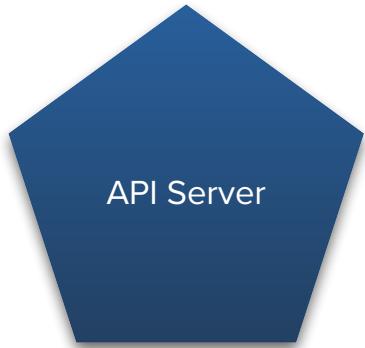
Not related to Kubernetes, some ideas:

- Write software without vulnerabilities (easy).
- Audit your code by third party (hehe).
- Scan your container images for known vulnerabilities (lot of choices). And truly fix them!
- So many things...





Solving - First issue: vulnerability in software





Second issue: credentials in container

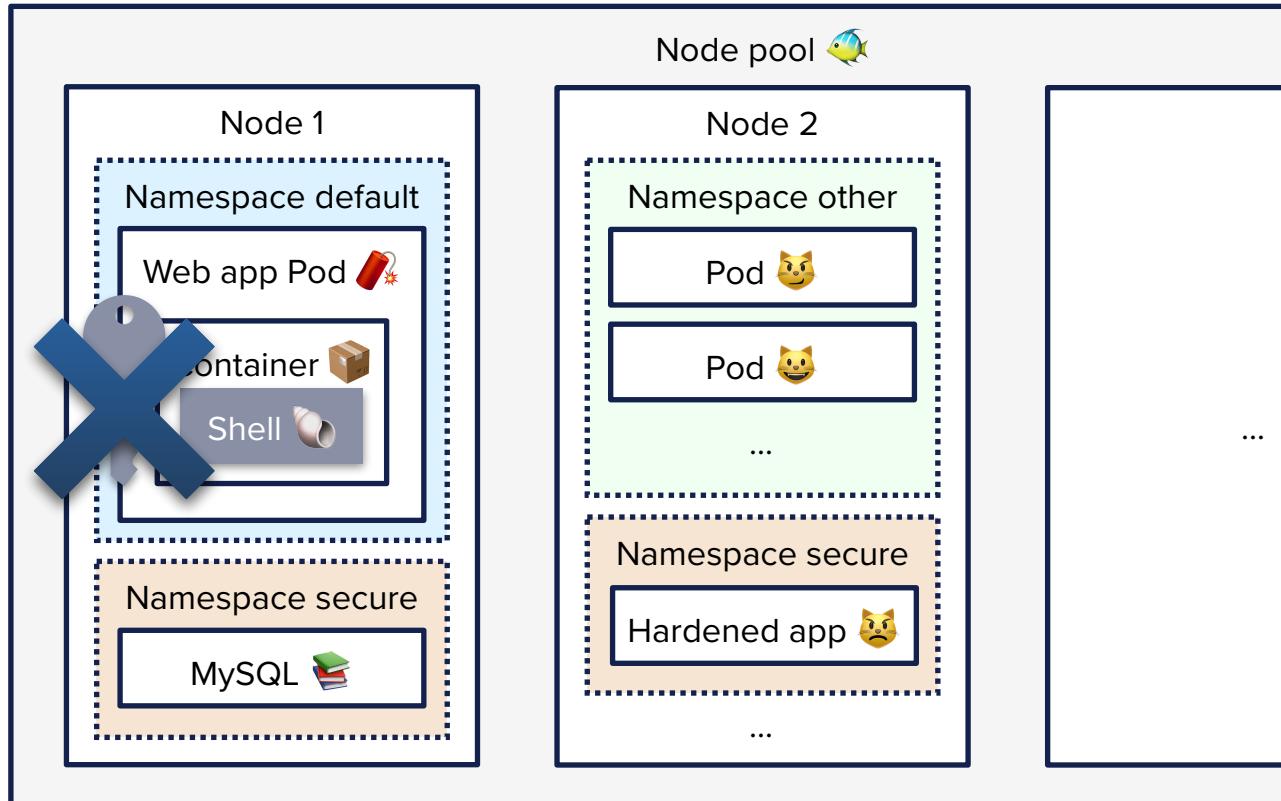
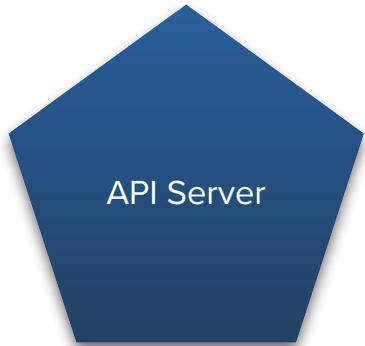


Be extra careful with rights and tokens

- Configure wisely your RBAC.
- Avoid roles with a lot of rights.
- Be extra careful with rights on resources like Pods or similar with create or update verbs.
- Avoid mounting service account token if possible.
- See <https://kubernetes.io/docs/concepts/security/rbac-good-practices/>.



Solving - Second issue: credentials in container





Third issue: no admission control

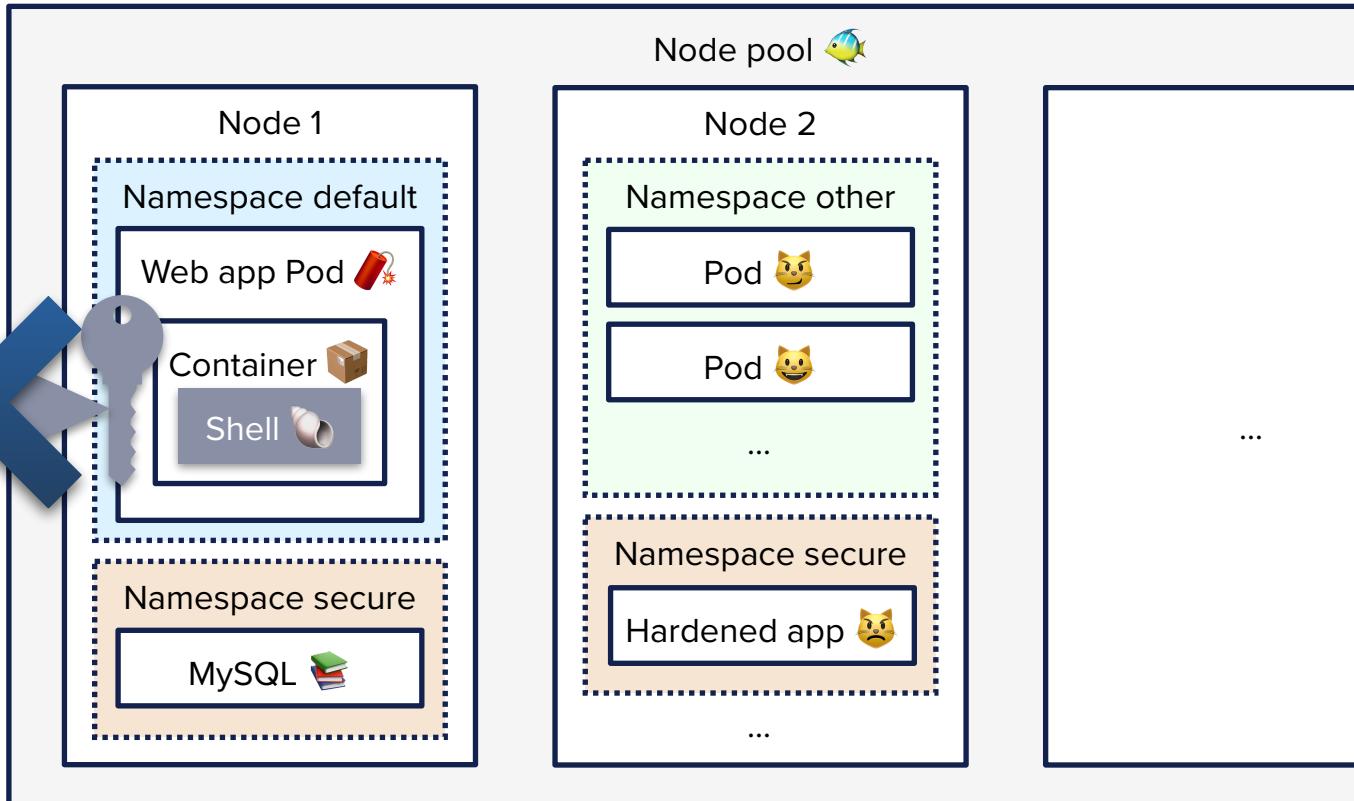
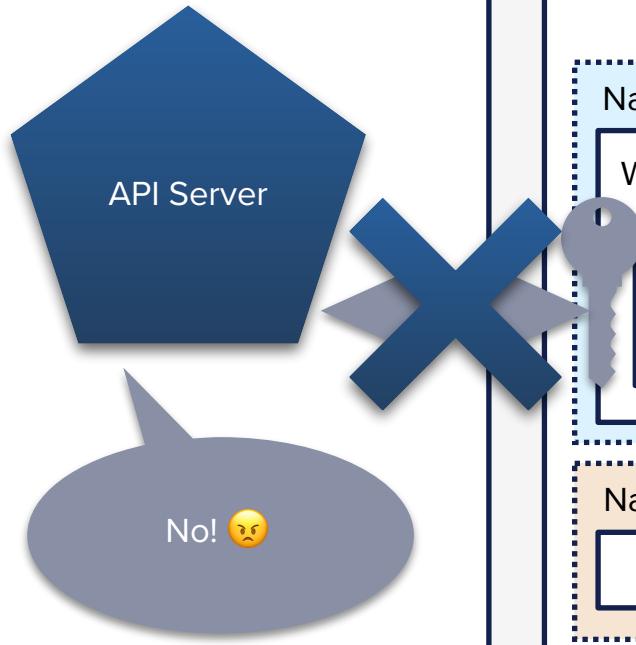
Kubernetes RBAC is not enough.

- RBAC is allow/disallow to create pod but this is the maximum granularity.
- Use admission controllers to be more granular.
- Integrated one: **PodSecurityPolicy** deprecated in 1.21 and removed in 1.25. Replacement **PodSecurity** is the new standard, in beta since last version, 1.23.
- Third party ones: Kyverno, OPA/Gatekeeper, etc.





Solving - Third issue: no admission control



If you want to experiment by yourself



[GitHub - quarkslab/kdigger](https://github.com/quarkslab/kdigger)

github.com/quarkslab/kdigger

Product Team Enterprise Explore Marketplace Pricing Search Sign in Sign up

Code Issues Pull requests Actions Security Insights

master 1 branch 4 tags Go to file Code About

mtardy Add a small message in the CHANGELOG c61ab3a 15 days ago 74 commits

commands Clarify arch is the builder and pull go ver from runtime 20 days ago

pkg Finish arm64 build properly by skipping empty gaps in sys... 15 days ago

.gitignore Add an unfinished arm64 build 15 days ago

CHANGELOG.md Add a small message in the CHANGELOG 15 days ago

LICENSE Update README and add LICENSE 10 months ago

Makefile Reorganize the release target in Makefile 15 days ago

README.md Update TOC in README 15 days ago

Vagrantfile Update Vagrantfiles and Makefile 3 months ago

go.mod Update to Go 1.18 3 months ago

go.sum Update to Go 1.18 3 months ago

main.go Update package path from mtardy to quarkslab on github ... 10 months ago

README.md

kdigger

kdigger, short for "Kubernetes digger", is a context discovery tool for Kubernetes penetration testing. This tool is a compilation of various plugins called buckets to facilitate pentesting Kubernetes from inside a pod.

Kubernetes focused container assessment and context discovery tool for penetration testing

kubernetes security tool containers pentest

Readme Apache-2.0 license 277 stars 11 watching 13 forks

Releases 4

v1.2.1 (Latest) 15 days ago + 3 releases

Contributors 2

mtardy Mahé 06kellyjac j-k

[GitHub - quarkslab/minik8s-ctf](https://github.com/quarkslab/minik8s-ctf)

github.com/quarkslab/minik8s-ctf

Product Team Enterprise Explore Marketplace Pricing Search Sign in Sign up

Code Issues Pull requests Actions Security Insights

master 1 branch 0 tags Go to file Code About

mtardy Update koolbox images to latest version with kdigger v1.2.1 8e2a224 5 days ago 11 commits

scripts Add global env vars for setup scripts 5 days ago

LICENSE Add LICENSE 10 months ago

README.md Update README.md 3 months ago

challenge-qits.yaml Update koolbox images to latest version with kdigger v1.2.1 5 days ago

challenge1.yaml Update koolbox images to latest version with kdigger v1.2.1 5 days ago

challenge2.yaml Update koolbox images to latest version with kdigger v1.2.1 5 days ago

challenge3.yaml Update koolbox images to latest version with kdigger v1.2.1 5 days ago

demo-challenge1.cast First commit 10 months ago

demo-challenge2.cast First commit 10 months ago

scripts.yaml Add Quarks In The Shell presentation challenge (#3) 3 months ago

setup.sh Add global env vars for setup scripts 5 days ago

solutions.md First commit 10 months ago

start.sh Add a setup script for macOS and simplify start.sh 3 months ago

README.md

Minik8s CTF

A beginner-friendly CTF about Kubernetes security.

kubernetes security ctf

Readme Apache-2.0 license 52 stars 6 watching 7 forks

Releases

No releases published

Packages

No packages published

Languages

Shell 100.0%

@mtardy_

29

If you want to experiment by yourself



kdigger: a Context Discovery Tool for Kubernetes

Date: Thu, 07 October 2021 By: Mahé Tardy Category: PenTest Tags: Kubernetes, Cloud, Tool

This article is an introduction to Kubernetes security through the presentation of a new context discovery tool. It was built in reaction to the capture the flag challenge of the Europe 2021 KubeCon Cloud-Native Security Day CTF. We open-sourced the tool, named kdigger, on GitHub.

I will quickly introduce Kubernetes, and then kdigger through a little CTF created to highlight its usage.

Quick introduction on Kubernetes

If you are already familiar with Kubernetes and its basic concepts, please feel free to skip this section.

To understand this introduction, you need some Kubernetes vocabulary, these definitions are extracted from the official Kubernetes glossary where you can find a lot more:

- **Pod:** The smallest and simplest Kubernetes object. A pod represents a set of running containers on your cluster.
- **Node:** A node is a VM or a physical machine in Kubernetes.

Kubernetes was released by Google as an open-source project in 2014. It was built with the 10 years of experience Google acquired by creating Borg, its main cluster manager in the early 2000s. It describes itself as “an open-source system for managing containerized applications across multiple hosts [and] provides basic mechanisms for deployment, maintenance, and scaling of applications”.

```
graph TD; subgraph KCP [Kubernetes Control Plane]; KCM1[kube-controller manager]; KCM2[cloud-controller manager]; end; KCM1 --> KCM2; KCM2 --> KAS[kube-api-server]; KAS --> Cloud((Cloud))
```



Recent new releases

- Added new plugins: **cgroup**, **node**, **apiresources**,
- Added new flag checks from */proc/self/status*
- New support for **Linux aarch64** (and Darwin x86_64)
- You can now install kdigger with **Nix!**





Kubernetes community
is working to improve
its **security**

New documentation on RBAC good practices (Mai 15)



 **kubernetes**

Documentation Kubernetes Blog Training Partners Community Case Studies Versions ▾ English ▾

Kubernetes Documentation / Concepts / Security / Role Based Access Control Good Practices

Role Based Access Control Good Practices

Kubernetes RBAC is a key security control to ensure that cluster users and workloads have only the access to resources required to execute their roles. It is important to ensure that, when designing permissions for cluster users, the cluster administrator understands the areas where privilege escalation could occur, to reduce the risk of excessive access leading to security incidents.

The good practices laid out here should be read in conjunction with the general [RBAC documentation](#).

General good practice

Least privilege

Ideally minimal RBAC rights should be assigned to users and service accounts. Only permissions explicitly required for their operation should be used. Whilst each cluster will be different, some general rules that can be applied are :

- Assign permissions at the namespace level where possible. Use RoleBindings as opposed to ClusterRoleBindings to give users rights only within a specific namespace.
- Avoid providing wildcard permissions when possible, especially to all resources. As [Kubernetes is an extensible system, providing wildcard access gives rights not just to all](#)

Edit this page
 Create child page
 Create an issue
 Print entire section

General good practice
Least privilege
Minimize distribution of privileged tokens
Hardening
Periodic review
Kubernetes RBAC - privilege escalation risks
Listing secrets
Workload creation
Persistent volume creation
Access to proxy subresource of Nodes
Escalate verb
Bind verb
Impersonate verb
CSRs and certificate issuing
Token request

New documentation on multi-tenancy (June 23)



The screenshot shows the Kubernetes Documentation website with the following details:

- Header:** Includes the Kubernetes logo, a search bar, and navigation links: Documentation (underlined), Kubernetes Blog, Training, Partners, Community, Case Studies, Versions ▾, and English ▾.
- Breadcrumbs:** Kubernetes Documentation / Concepts / Security / Multi-tenancy
- Section Header:**

Multi-tenancy
- Text:** This page provides an overview of available configuration options and best practices for cluster multi-tenancy.
Sharing clusters saves costs and simplifies administration. However, sharing clusters also presents challenges such as security, fairness, and managing *noisy neighbors*.
Clusters can be shared in many ways. In some cases, different applications may run in the same cluster. In other cases, multiple instances of the same application may run in the same cluster, one for each end user. All these types of sharing are frequently described using the umbrella term *multi-tenancy*.
While Kubernetes does not have first-class concepts of end users or tenants, it provides several features to help manage different tenancy requirements. These are discussed below.
- Section Header:**

Use cases
- Text:** The first step to determining how to share your cluster is understanding your use case, so you can evaluate the patterns and tools available. In general, multi-tenancy in Kubernetes clusters falls into two broad categories, though many variations and hybrids are also possible.
- Section Header:**

Multiple teams
- Text:** A common form of multi-tenancy is to share a cluster between multiple teams within an organization. This is often done using namespaces, which provide a way to isolate different teams' workloads from each other.
- Right sidebar:** Includes links for editing the page, creating child pages, and printing sections, as well as a list of related topics: Use cases, Multiple teams, Multiple customers, Terminology, Tenants, Isolation, Control plane isolation, Namespaces, Access controls, Quotas, Data Plane isolation, Network isolation, Storage isolation, Sandboxing containers, Node isolation, and Additional Considerations.

New documentation on security checklist



[kubernetes / website](#) Public

Edit Pins Watch 201 Fork 11.2k Star 3.3k

Code Issues 553 Pull requests 190 Actions Projects 4 Wiki Security Insights

Add a security checklist for clusters #33992

I Open mtardy wants to merge 2 commits into [kubernetes:main](#) from [mtardy:security-checklist](#)

Conversation 159 Commits 2 Checks 4 Files changed 1

mtardy commented on 27 May

Addresses issue [Create a security checklist for deploying a cluster #28](#) in k/sig-security repository.

Draft for the security checklist guide from the collaborative document with @savitharaghunathan, @Skybound1 and @p4ck3t0. Thanks everyone 😊!

It is still missing some stuff and things need to be addressed before merging:

- Some consistency issues, some topics are written with paragraphs and others via lists style.
- It feels that more could be done on the "Image" part with all the supply chain enthusiasm we are getting from the community!
- I took the liberty to comment the paragraph in Authentication & Authorization because of the complete guide on RBAC.

If anyone has ideas about topics to add, don't hesitate to participate.

k8s-ci-bot added do-not-merge/work-in-progress, cncf-cla: yes, size/l. labels on 27 May

k8s-ci-bot commented on 27 May

[APPROVALNOTIFIER] This PR is NOT APPROVED

This pull-request has been approved by:

Reviewers

- raesene
- tengqm
- callynse
- Skybound1
- NissesSenap
- divya-mohan0209
- sftim
- reylejano
- rschossler
- p4ck3t0
- bradtopol
- jimangel
- rajeshdeshpande02

Assignees

Still in progress? Convert to draft

Thank you!



- Kubernetes focused **container** assessment and **context discovery** tool for **penetration testing**.
- **kdigger** is open-source and available at
<https://github.com/quarkslab/kdigger>.
- **minik8s-ctf** is another open-source CTF project to directly experiment with **kdigger**
<https://github.com/quarkslab/minik8s-ctf>.