



NSA/CISA Kubernetes Hardening Guidance and Beyond

Securing Kubernetes Container Platforms in 2022

2022-11-23



PhD & Senior Cloud Architect @ [Lars Larsson](#)
[Elastisys](#)



Reality check

Containers live in a world where...

- [56% of companies](#) are **not vulnerability scanning**
- [~70% of companies](#) will run containerized workloads in 2023
- [92% of Kubernetes-running companies](#) have had a security incident in the **last year**
- **Misconfiguration** makes up [59% of Kubernetes security incidents](#)



Uncomfortable question

What permissions have you given me
in your Kubernetes cluster if I manage
to hack into your application?



whoami

Lars Larsson

- Senior cloud architect at [Elastisys](#)
- Worked in cloud computing since 2008
- PhD in CS for research on cloud resource management
- Software developer, DevOps engineer, distributed systems

[Connect on LinkedIn](#): security, DevOps, Kubernetes,
compassionate leadership



NSA/CISA Kubernetes Hardening Guidance

Tech report from August 2021 (1.0) and [updated in September 2022](#) (1.2)

- Focused on Kubernetes **itself**
 - Detailed discussion of security-related configuration
 - Example code
- Mentions other security software (in passing)
- Focuses mainly on **hardening**, not *security as a process*

1.

Scan containers and Pods for vulnerabilities or misconfigurations

- Container images are **immutable**
 - Insecure code stays in time capsule
- Component stability: infrequent updates → worse security
- Scan images for known vulnerabilities
- Report: using an Admission Controller

Difficulty



Impact



 **Lars adds**

- Daily scan all images that are in active use!

2.

Run containers and Pods with the least privileges possible

- 🤖 Containers run as “root” by default 🤖
- Container file systems: read-only until need arises
- Use most restrictive Pod Security Policies (<= v1.21) or Pod Security Standard (>= 1.22)
- Avoid handing Default Service Account to Pods
- Limit kernel interactions using SELinux, AppArmor, or gVisor-like sandboxes

Difficulty 

Impact 

👤 Lars adds

- **Also** encode your policies for automatic enforcement via Open Policy Agent

3.

Use network separation for compromise damage control

- Separate control plane and worker node networks
- Default settings allow all Pods to network with all other Pods
 - Security is only as strong as the weakest point!
- Network Policies are Kubernetes-aware firewall rules
 - Specify rules for IP blocks or Kubernetes objects
 - Allow **only** “backend” to connect to “database” but **nothing else**

Difficulty



Impact



4.

Protect confidentiality: firewalls and encryption

- Restrict access to Kubernetes core components
 - API server, etcd, Controller Managers
- Network traffic in Kubernetes clusters is unencrypted by default

Lars adds

- Use a networking provider with transparent encryption, e.g., [Calico with WireGuard support](#) or consider a service mesh

Difficulty



Impact



5.

Limit attack surface: authn, authz, RBAC

- Enable authentication, authorization, and role-based access control

Lars adds

- Disable the **perpetual** admin token created during installation
- OpenID Connect for user and **group membership**
- **Disable anonymous access**
- **Restrict permissions** as much as possible with RBAC

Difficulty



Impact



6.

Audit logging, log auditing

- Capture all logs from the entire environment, collect into single system
- All API calls can be logged for auditing purposes
 - Log at appropriate levels to avoid logging e.g. Secrets
 - Creates a huge amount of logs!
- Use an automated system for processing audit logs

Difficulty 
Impact 

 **Lars adds**

- [Falco](#) as a simple [Security Incident and Event Management \(SIEM\)](#) system together with centralized logging, e.g., [OpenSearch](#)

7.

Periodically scan, review, and patch Kubernetes

- Kubernetes has a new release ~3 times per year
 - N-3 security updates support (current and the two previous ones)
- Security features are typically opt-in, rather than opt-out
 - You need to opt-in as soon as possible
- Automated testing can help find insecure (default) settings





Are automated vulnerability tools sufficient?

- [Kubescape, kube-bench](#)
 - Investigate Kubernetes API (kubescape) or control plane host
- Low-hanging fruit of vulnerability scanning

Lars adds

- You **must do this** to not scream “insecure cluster over here!”
- **Limited** in what they can investigate, and always will be
- Encryption at rest storage, firewall rules, security policies encoded in other systems than Kubernetes, underlying operating system, third-party software...



8 practical advice from Lars

Beyond the NSA/CISA recommendations



1.

Prevent misconfiguration, don't just check for it

- [2/3 of all insider incidents are due to negligence](#)
- RBAC is great, but limited in what it can express:
 - “Lars” allowed to “modify configuration” in the “production” environment
 - ...but is he allowed to make any configuration change he pleases?
- [Open Policy Agent](#) to the rescue (again!)
 - [Library](#) and other [third-party rules](#) as inspiration

Difficulty 

Impact 

2.

Any permission given to an application is also given to bad actors

- Hacked applications have all the permissions that the application usually has
 - Third-party SaaS integrations
 - VPN-connected back-office locations
 - Databases
- **Always** restrict your app components as much as possible



3.

Keep cloud resources, specifically, in mind, too.

- Various Controllers and Operators in the community offer cloud integrations.
 - How seriously do they take cloud security?
- **Reject** ones without configurable/restrictive permissions

Difficulty



Impact



4.

Does your app **unintentionally** have permissions in your cloud?

- **Beware** of “instance profiles” that your cluster VMs may have ability to modify
 - DNS records,
 - autoscaling groups,
 - load balancers...
- ...because all applications can also get those permissions!
 - Just call the cloud’s metadata service + get token w/ permissions
 - Applications are also “the VM” to the cloud

Added in 1.1

Difficulty



Impact



5.

✨ Added in 1.2

Do this daily!

Regularly scan all your deployed container images, not just when they are new.

- To get up to date security scans, *you just need to:*
 - loop through all your Pods that are deployed,
 - determine which container images are in active use, and
 - scan those images.
- More secure than “scan on push” or “scan on deployment”
- “Zero trust”: scan, sign, verify, enforce policies with private registries

Difficulty 

Impact 

6.

Foster a security-first mindset

Regularly have **your own staff** security test your entire system.

- Building is hard, breaking is easy (and fun!)
- Your engineers have access to source code, hacker's don't
- Let your engineers try to break your application
- Better if they find errors, than if hackers do!

Difficulty



Impact



7.

Disaster Recovery != "backups"

Have a **Disaster Recovery (DR)** plan, and actually practice it.

- Disaster could be "entire cloud region outage"...
- ...or "we need to go back in time to five hours ago, before this attack started"

How quickly can you destroy your entire infrastructure and get it back again?

Difficulty 

Impact 

8.

IDS & SIEM

Use an Intrusion Detection System **and** a Security Information and Event Management.

- Intrusion Detection System (IDS) verifies that applications behave “normally”
- Security Information and Event Management (SIEM) searches through logging systems to find and flag abnormal events
- Could be false positives, but could also be indications of incidents!
- [Falco](#) is an IDS and can also be a simple SIEM

Added in 1.1

Difficulty 

Impact 

Summary

Kubernetes is neither safe by default, nor by itself.

- Restrict access (network, users, machines) and privileges
- Periodically use tools to assess current security practices
- Prevent misconfiguration, don't just check after the fact
- Cloud resources and permissions: be mindful!
- Security-conscious engineering culture
- Disaster Recovery applies also to security breaches





Uncomfortable question

What permissions have you given me
in your Kubernetes cluster if I manage
to hack into your application?



Appropriate Answer

No more than **absolutely** needed!

And you will see that I am there, because you have automated systems that both limit what I can do, and raise an alert when I make the application behave in ways it doesn't normally do.



[Free Guide: How to security
harden Kubernetes in 2022](#)

(this talk's article)



[Free Guide: How to Operate
a Secure Kubernetes
Platform](#)



Securing Kubernetes Container Platforms in 2022

Do you have a question? Fire away!

or via [linkedin.com/in/larsson/](https://www.linkedin.com/in/larsson/)

