



WHITEPAPER

CNAPP & Policy-as-Code: Complementary, Yet Distinctly Unique

From CNAPP signal overload to platform-enforced guardrails

EXECUTIVE SUMMARY

Visibility alone is no longer enough

Cloud Native Application Protection Platforms (CNAPPs) such as Wiz have become essential for modern cloud security teams. They provide deep visibility into cloud risk, runtime threats and attack paths across Kubernetes and cloud environments.

But as Kubernetes adoption accelerates, misconfigurations remain the leading cause of cloud security incidents¹, and most of them originate **before runtime**: in platform configuration, CI/CD pipelines and day-to-day operational changes. A CNAPP finds them after they exist. Someone still has to fix them, usually by hand and usually more than once.

This is where Nirmata and Kyverno complement CNAPP:

- **CNAPPs** excel at security discovery, exposure analysis and runtime risk detection. They are typically owned by the CISO organization.
- **Nirmata, powered by Kyverno**, provides proactive, highly flexible platform-level controls using Policy-as-Code (PaC), owned by platform engineering and DevSecOps teams.

Nirmata enforces platform guardrails that stop risky configurations from being created in the first place. That reduces CNAPP noise and lets security teams focus on real threats.

CNAPP ANSWERS

“Where are we exposed right now?”

NIRMATA ANSWERS

“How do we prevent issues and enforce everywhere?”

Key takeaways

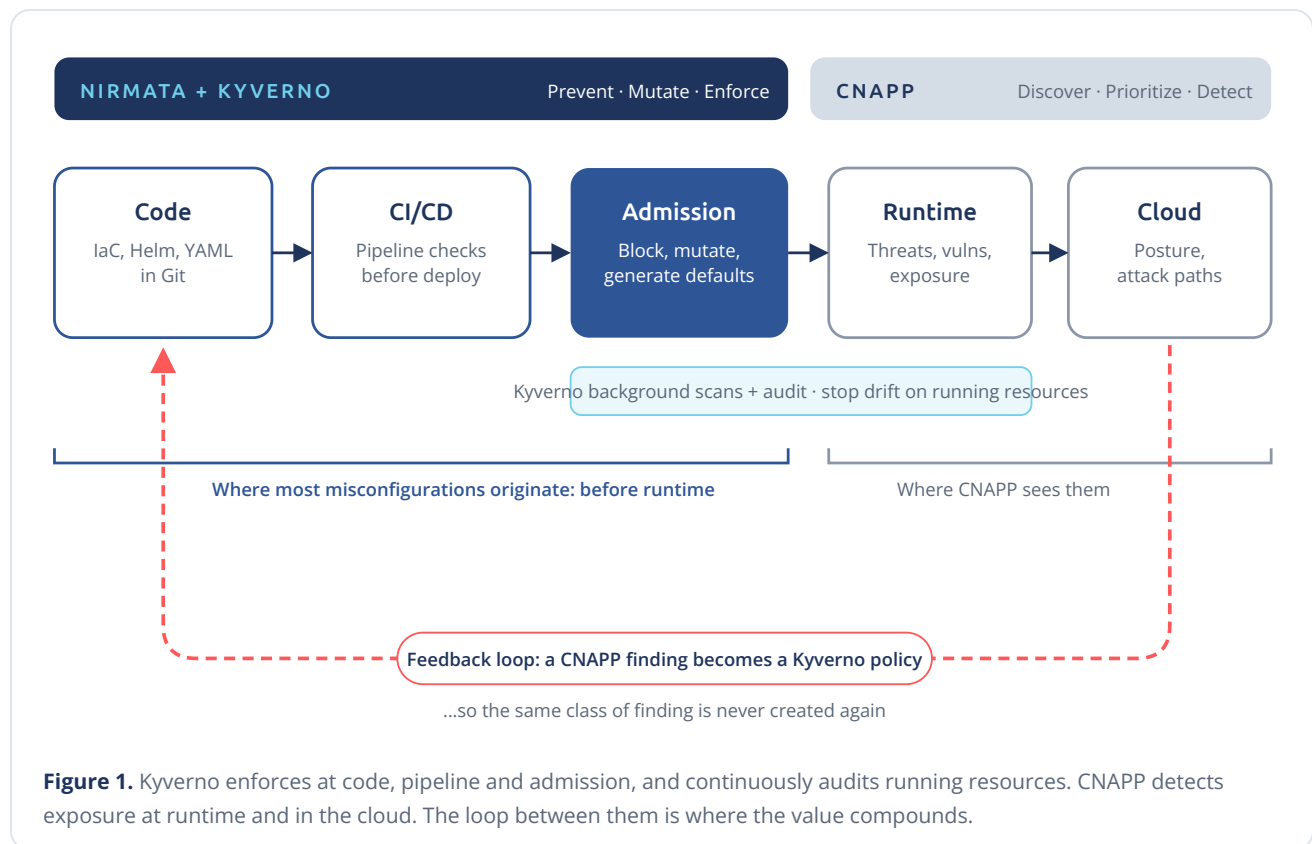
- **Different layers, different owners.** CNAPP detects and prioritizes for security. Kyverno prevents and enforces for the platform.
- **Prevention removes whole classes of findings.** One policy at admission replaces hundreds of recurring tickets.
- **Nirmata amplifies CNAPP.** Fewer hygiene findings means a higher signal-to-noise ratio for the alerts that matter.

¹ “Why Are Misconfigurations Still The Top Cause Of Cloud Breaches?” Forbes Technology Council, September 23, 2025. [forbes.com](https://www.forbes.com)

WHERE EACH LAYER ACTS

Kyverno prevents and enforces while CNAPP detects

Every Kubernetes resource moves through the same lifecycle: it is written, built and tested, admitted to a cluster, and then runs. Kyverno acts on the left side of that lifecycle, at the points where a bad configuration can still be stopped or corrected. A CNAPP acts on the right, where it sees what is actually running and how exposed it is.



Why the loop matters

Without the loop, every CNAPP finding becomes a ticket, and every ticket becomes a manual fix that can be reintroduced by the next deployment. With the loop, security identifies a risk once, the platform team codifies it once, and the guardrail applies everywhere from that point forward: in every pipeline, every cluster and every namespace.

THE CHALLENGE

CNAPP alone is not enough for platform teams

CNAPPs are designed primarily for security teams, and they are very good at what they do:

Runtime vulnerability assessment

Cloud security posture management

Threat detection

Risk prioritization

Attack path analysis

Platform and DevOps teams face a different problem. Their job is to give hundreds of developers a self-service platform that is safe by default, without becoming a ticket queue. A list of findings, however well prioritized, does not do that on its own.

Where CNAPPs fall short for platform engineering

Limitation	Why it matters to the platform team
Built for detection and prioritization	Cannot enforce across clusters and pipelines with safe defaults, mutation and generation, or consistent exception workflows.
Enforcement scoped to security	Cannot encode platform standards, automate safe defaults, or manage exceptions the way a policy system does.
Policies cover security only	Does not address the operational, cost and compliance standards across the full developer platform SDLC.
Not developer-facing	Developers still rely on tickets and manual remediation instead of getting feedback in the pipeline.
Weak drift prevention	Identifies drift but does not stop it from recurring with the next deployment.

The result: CNAPP insights become alerts and tickets that must be translated into platform changes by hand, which increases MTTR and operational toil.

THE APPROACH

Nirmata + Kyverno: proactive platform governance

Nirmata embeds security, compliance and operational best practices directly into the Kubernetes platform using Kyverno, the CNCF-graduated policy engine Nirmata created. Guardrails are always on, everywhere, and far less dependent on human review. **Nirmata becomes the policy system of record for the platform.**

What Nirmata does differently

- 01** **Shift-Down Security.** Common security and operational controls are enforced by the platform itself, not bolted on later.
- 02** **Guardrails that eliminate classes of findings** by preventing misconfigurations at the source.
- 03** **Inline feedback in CI and at admission**, with clear policy messages and automated defaults.
- 04** **Policy as a platform contract** across clusters, namespaces and pipelines.
- 05** **Drift stops** through enforced controls, generated baselines and exception workflows.

Key capabilities

■ Policy as Code

Kyverno policies for Kubernetes and cloud-native resources, written in YAML or CEL.

■ CI/CD policy checks

Stop misconfigurations in the pipeline, before they deploy.

■ Continuous audits

Background scanning and drift detection on running resources.

■ AI-assisted authoring

AI-assisted policy creation and remediation.

■ Policy lifecycle

Manage policies from authoring and testing through deprecation.

■ Multi-team workflows

Platform teams define standards; application teams consume them.

■ Exception management

Time-bound, scoped exceptions with full audit trails.

■ Consistent everywhere

The same policies across pipelines, clusters and cloud.

IN PRACTICE

From finding to guardrail

Consider a common case. The CNAPP flags 43 workloads across 25 clusters running privileged containers. Without a policy layer, that becomes 43 tickets routed to the owning application teams, and the finding returns the next time someone copies an old manifest. With Nirmata, it becomes a single guardrail.

01 · CNAPP

Detect

Security sees privileged pods, ranked by exposure and attack path.

02 · PLATFORM

Codify

The platform team adopts a Kyverno ValidatingPolicy, AI-assisted or from the policy library.

03 · NIRMATA

Enforce

Audit first, then enforce in CI and at admission. Exceptions are scoped and logged.

04 · BOTH

Verify

Background scans clear existing violations. The CNAPP finding class trends to zero.

```
apiVersion: policies.kyverno.io/v1
kind: ValidatingPolicy
metadata:
  name: disallow-privileged-containers
spec:
  validationActions: [Deny]           # start with [Audit], then switch
  evaluation:
    background:
      enabled: true                   # also audits resources already running
  matchConstraints:
    resourceRules:
      - apiGroups: [""]
        apiVersions: [v1]
        operations: [CREATE, UPDATE]
        resources: [pods]
  variables:
    - name: allContainers
      expression: >-
        object.spec.containers +
        object.spec.?initContainers.orValue([]) +
        object.spec.?ephemeralContainers.orValue([])
  validations:
    - expression: >-
        variables.allContainers.all(c,
          c.?securityContext.?privileged.orValue(false) == false)
      message: >-
        Privileged mode is not allowed. Remove securityContext.privileged
        or request an exception from the platform team.
```

A Kyverno ValidatingPolicy written in CEL, enforcing the Pod Security Standards (Baseline) control on privileged containers. The developer sees the message above in the pipeline or at `kubectl apply`, not in a ticket a week later.

OPERATING MODEL

Who does what

The loop works because each team keeps doing the job it is best placed to do. Security does not have to become a platform team, and the platform team does not have to become a security operations center.

Security

CISO ORGANIZATION

- Runs the CNAPP and owns risk priorities
- Identifies which finding classes recur most
- Approves high-risk exceptions

Platform

PLATFORM ENGINEERING / DEVSECOPS

- Codifies standards as Kyverno policy
- Rolls out via GitOps, audit before enforce
- Owns safe defaults and exception workflows

Developers

APPLICATION TEAMS

- Get inline feedback in CI and at admission
- Inherit safe defaults automatically
- Request scoped exceptions, not tickets

What changes

Before: CNAPP alone	After: CNAPP + Nirmata
One ticket per workload, routed by hand	One policy, applied to every cluster through GitOps
The finding reappears on the next deploy	The violation is blocked before it reaches the cluster
Developers learn about problems days later	Developers see a clear policy message in the pipeline
Exceptions live in email and spreadsheets	Exceptions are scoped, time-bound and auditable
CNAPP queues are full of recurring hygiene issues	CNAPP queues surface the findings that need investigation

BETTER TOGETHER

Different layers of the security stack

	CNAPP Wiz, Prisma Cloud, Sysdig, etc.	Nirmata + Kyverno
CORE JOB	Discovers cloud risk	Prevents risky configurations from being created, with guardrails and safe defaults
ANALYSIS	Analyzes attack paths	Enforces platform guardrails
AT DEPLOY TIME	Detects runtime exposure after deploy	Blocks violations at admission; mutates and generates safe defaults
OWNER	Security team	Platform team
MODE	Detection and investigation	Prevention and enforcement

Capabilities unique to the policy layer

GitOps-based policy deployment

Open reporting that integrates with platform tools

Background scanning

Low-latency admission control

Mutation and generation for automation

Nirmata amplifies CNAPP value

Combined, the two give organizations a balanced, scalable and developer-friendly security model. CNAPP handles security discovery and runtime protection. Nirmata prevents risky configuration paths and standardizes safe defaults.

Because Nirmata eliminates misconfigurations before they reach production, the CNAPP reports fewer findings tied to configuration drift and policy violations. That reduction cascades: security teams see fewer alerts and tickets, platform teams do less manual remediation, and tools like Wiz can surface the higher-signal findings that truly warrant investigation.

Nirmata doesn't compete with CNAPP. It amplifies it, by improving the signal-to-noise ratio so security teams can focus on real risk instead of recurring hygiene issues.

GETTING STARTED

Four steps to close the loop

- 01 Pick the recurring findings.** Pull the top configuration-related finding classes from your CNAPP: privileged containers, missing resource limits, images from untrusted registries, host path mounts.
- 02 Map each one to a policy.** Most have a ready-made Kyverno policy in the open-source library or in Nirmata's curated policy sets; the rest can be drafted with AI assistance.
- 03 Roll out in audit mode.** Deploy through GitOps, review policy reports with application teams, and agree on exceptions before anything is blocked.
- 04 Enforce and measure.** Switch to enforce in CI and at admission, then track the matching CNAPP finding classes over time. They should trend toward zero.

This is not security vs. platform engineering.

It's security and platform engineering, working together.

About Nirmata

Nirmata created Kyverno, the CNCF-graduated policy engine for Kubernetes, and maintains it as Apache 2.0 open source. Nirmata Control Hub and Nirmata Enterprise for Kyverno enforce security, compliance and operational policy across four control points (pipeline, admission, gateway and runtime) with one policy language and one audit trail, with no fork and no lock-in.

[Request a demo · nirmata.com/request-a-demo](https://nirmata.com/request-a-demo)

[Explore Kyverno · kyverno.io](https://kyverno.io)