Container Security Buyer's Guide

Identify vulnerabilities, prioritize risks, and stop attacks on your environment.

Table of Contents:

Containers, Kubernetes, and cloud redefine computing 3

Why container security matters (and why it's challenging) 4

Key security risks for container environments 5

- Vulnerabilities in container images 5
- Misconfigurations in third-party libraries 5
- Misconfigurations due to the interplay between host, cloud, container, and Kubernetes cluster 6
- Excessive container privileges 6
- Sensitive data in the container writable layer 6

Where container security tools fall short 6

- Deployment models 7
- 3 essential functions a container security/KSPM solution must provide 7

Extending container security across the DevOps lifecycle—from shift-left to runtime 9

- Code 9
- Build 9
- Deploy 10
- Run 10

Essential function checklist to address key container security use cases 11

- Code 11
- Build 11
- Deploy 11
- Run 12
- Explore industry-leading container security/KSPM 12

To meet accelerating and unpredictable customer expectations, business needs, and market dynamics, companies have turned to a new model for application development. Cloud-native app development can have a transformative impact on business agility, making it possible to innovate, pivot, and respond to changing needs faster than ever. At the same time, this new application architecture also calls for new ways of thinking about security for cloud-native applications.

In this guide, we'll talk about what's driving the adoption of cloud-native application development, why it can increase risk, and what's needed to close the security gaps it can introduce.

Containers, Kubernetes, and cloud redefine computing

Just as cloud resources have reshaped modern computing environments, cloud-native methods have redefined application development. Instead of being developed monolithically, cloud-native apps are assembled from loosely coupled, task-specific microservices connected via APIs. Independently deployed, updated, and scaled, microservices allow greater resiliency and agility by making changes simpler, faster, and lower risk.

Containers, a lightweight form of operating system virtualization, are used to encapsulate both the microservices making up a cloud-native app and the related elements it needs to run, such as system libraries, system settings, and other dependencies. Compact, portable, scalable, and standardized, containers run the same way across clouds and operating systems, making it easy to move and deploy them across on-demand cloud-based development, test, and production environments. For developers, containers are ideally suited to Agile and DevOps practices—and essential for meeting the fast-paced demands of digital business.

In fact, containers have quickly become a cornerstone of the cloud era. In a recent survey by the Cloud-Native Computing Foundation (CNCF), 44 percent of respondents were already using containers for nearly all applications and business segments, while 35 percent were doing so for at least a few production applications; another 9 percent were piloting or actively evaluating containerization.

As the use of containers has grown, so has the need for simple ways to work with them. Kubernetes, an open-source container orchestration system, has become the mainstream choice for automating container deployment, scaling, and management.

A versatile, extensible, and highly portable environment, building and running cloud-native apps, Kubernetes:

- Handles the process of initiating, pausing, or rolling back container deployments
- Automatically exposes containers to one another and to networked services and resources
- Ensures that containerized applications can scale up and down quickly, automatically, and reliably
- Automatically monitors container health and performance, and restarts or replaces failing containers
- Makes it easy to move apps across hybrid and multi-cloud environments

According to CNCF, 96 percent of organizations are currently using or evaluating Kubernetes, including 5.6 million developers. This raises a critical question: are they doing it securely?

Why container security matters (and why it's challenging)

Containers offer tremendous business value, but can also pose challenges for security. To name a few:

- A lack of complete visibility and understanding of the containerized environment – The configuration of container orchestration and cloud environments is a complex, multi-step process that can inadvertently result in "toxic combinations"—an accumulation of individual risk factors that combine to offer an attack path to critical resources.

- Security vulnerabilities within the container itself – As with any development approach, cloud-native apps can include security vulnerabilities in the libraries and software packages distributed in a container.

Fragmented scanning and monitoring – Siloed tools for cloud-centric, cluster-centric, and configuration-centric vulnerability scanning can generate a tremendous volume of alerts with no guidance for prioritization—while also missing key threats.

The flexibility of containers can raise security issues as well. They can do everything from mounting volumes and directories to disabling security features, and even run as root under the control of a hacker in a "container breakout" scenario where container isolation mechanisms are bypassed and additional privileges are obtained on the host.

And consider that 90 percent of Kubernetes users leverage the managed Kubernetes services offered by cloud providers and running in the cloud, where security has always been notoriously challenging. As environments grow more complex, encompassing new workloads, architectures, roles, and users, it becomes painfully difficult to answer questions like "which of our databases are exposed to the internet?"

In fact, CNCF reports that the more widely companies use containers, the more likely they are to call security their top challenge with containers. Clearly, securing containers and Kubernetes environments is a top priority for every modern business. To protect your organization from threats, you need to be able to identify vulnerabilities and misconfigurations, Internet-facing containers, excessive permissions, and exposed secrets to proactively remove container risk and stop attack paths into your environments.

Key security risks for container environments

As organizations work to secure their Kubernetes and container environments, the following risks are especially important to address.

Vulnerabilities in container images

Image vulnerabilities are a common factor in container risk, whether resulting from an insecure library or imported dependency, or a threat introduced through a breach in the development environment. To keep such vulnerabilities from being introduced into the live environment, organizations need to scan images before they are used to create individual containers.

Misconfigurations in third-party libraries

In a complex cloud environment spanning multiple platforms, diverse architectures, and thousands of different applications and services, it's all too easy for the many third-party libraries in use to introduce bugs and other security vulnerabilities into the container environment.

Misconfigurations due to the interplay between host, cloud, container, and Kubernetes cluster

Overlapping layers of configurations, networks, and identities across the distributed c loud-native technology stack can easily lead to errors and security gaps such as accidentally exposing internal services. Organizations need a clear understanding of how these elements are connected to each other, to the corporate network, and to the internet, so they can identify inadvertent risks.

Excessive container privileges

A privileged container can allow development teams to access resources on the host, but if breached, it can do the same for a malicious actor. To prevent attackers from abusing root access to find and exploit vulnerabilities, it's crucial to limit and monitor container privileges as closely as possible.

Sensitive data in the container writable layer

Each container uses its own writable container layer to store changes, including the addition or modification of data. If someone gains illicit or unauthorized access to the container, they can make changes and additions of their own—and might even find secrets or other sensitive data being stored within the layer. It's important to be able to scan the writable layer to make sure that no sensitive data is being exposed to risk in this way.

Where container security tools fall short

Solving container security isn't easy. Many of the tools currently available to secure container environments have critical shortcomings:

- **Over-dependence on agents.** While most traditional solutions rely on the installation of an agent on the running resource. This can leave blind spots in coverage as well as friction with developers, who often see agents as a risk to stability and performance. An API-based approach could accomplish the same purpose in a more cloud-native manner.
- **Lacking the context needed to understand risk.** Siloed and limited container security tools can't correlate all three types of risk—container, Kubernetes, and cloud—to provide security teams with the insight they need to prioritize issues. For full understanding of risk, a solution needs to provide visibility into how systems and resources are configured, how they connect to the network and the internet, which identities can access them, and the permissions associated with those identities.

- **Focusing narrowly on production, not development.** When cloud-native apps are scanned for risk only after they've been built, security becomes reactive—stopping deployment rather than making it possible. To change security from a blocker to an enabler, security and development teams need to work together proactively to embed risk reduction throughout the development pipeline. This kind of shift-left thinking is as critical for container security as it is for any other aspect of development.

As security teams work to gain visibility into the containers in their environment and the risks they might be introducing, they also need to avoid undermining the velocity and agility containers are supposed to enable. Slow-moving preventive security processes can slow the continuous integration/continuous deployment (CI/CD) pipeline, increasing friction between security and development teams. Developers implementing high-velocity app or service delivery pipelines need a way to innovate with well-defined security guard-rails and a shared understanding of process, tooling, and policy.

Facing these requirements, what organizations need is a single solution for container security and Kubernetes security posture management (KSPM) that assesses risk across the full container stack, provides context and insight, and gives developers the tools to fix issues earlier and faster.

Deployment models

To be effective, a container security/KSPM solution must be able to discover and scan containers, hosts, and clusters across cloud-managed and self-managed Kubernetes environments, including serverless containers such as Fargate ECS as well as standalone containers running on VMs. Rather than requiring the installation of an agent, the solution should be easily deployable using an API connector and a single cloud role to scan the entire cloud and Kubernetes environment.

3 essential functions a container security/KSPM solution must provide

1. Enable continuous visibility

Dynamic, constantly changing Kubernetes environments are extremely hard to keep track of, as development teams spin up new clusters and workloads daily. Security teams need instant visibility into exactly what the environment looks like at any given moment.

2. Provide context

With different tools for monitoring workloads, entitlements, compliance, vulnerabilities, and more, each generating its own alerts, security teams can lose important signals among the noise. A container security/KSPM solution must provide a clear understanding of risk to prioritize a response, or to convince developers that it's important to do so.

3. Show risk at every level of the stack
To understand the risks associated with a container, it's not enough to focus solely on the container itself. You also have to consider the network, the entitlements in place, and the cloud environment as well.

Containers – Security teams need to make sure that containers don't contain vulnerabilities, and that they haven't been misconfigured to run on their host with permissions that are too powerful. Key requirements include the ability to:

- Validate container configuration and image at build time
- Sign container images
- Know where containers are running
- Identify running containers with vulnerabilities, malware, or secrets
- See which layer is impacted by a vulnerability.

Container orchestrators such as Kubernetes or AWS ECS – It's also essential to gain visibility into configurations across the network and identification layers used by containers. After all, the wrong configuration can expose containers to the Internet or grant excessive permissions in the cluster—or in the cloud itself. Key requirements include the ability to:

- Validate security policy violations before deployment
- Identify containers with over-privileged permissions on the orchestrator
- Block unwanted container images or bad configurations
- Understand cluster behavior

Cloud – In AWS, most containers are orchestrated by either Kubernetes or ECS. A successful attacker can compromise a container in an EC2 instance or another cloud service, and then laterally move to resources inside these clusters to use compute resources or access data. Key requirements include the ability to:

- Gain complete visibility and context on cloud layers (configuration, network, and identities)
- Identify over-privileged accounts
- Identity possible lateral movement from an attacked cloud service to a container

Extending container security across the DevOps lifecycle—from shift-left to runtime

Container security is a lifecycle, not a moment in time. It's critical to apply in a shift-left context, building risk reduction throughout the development process, but it's equally important in runtime, enabling security teams to detect vulnerabilities and threats in production, support audits, and perform forensics. Accordingly, a container security/KSPM tool must be designed to address every stage of DevOps:

- Code
- Build
- Deploy
- Run

Code

A full-lifecycle approach makes it possible to apply container security as an integral part of the CI/CD pipeline, including:

- Scanning Infrastructure-as-code (IaC) files across Dockerfiles, Kubernetes YAML manifests and Helm charts for misconfigurations and security risks
- Automatically assessing one security policy across the entire cloud and Kubernetes development lifecycle
- Validating compliance with security policy before container images are deployed in the cluster—instead of waiting until they're in production.

In this way, development teams can work autonomously to prevent security risks with an easy-to-use tool to secure container images and detect vulnerabilities and exposed secrets as part of their usual CI/CD pipeline. Security teams can avoid slowing the development process while ensuring that uniform security policies are being maintained across the cloud and container development lifecycle, from the build stage to the registry, to the runtime environment.

Build

As cloud-native apps reach the build stage, a full-lifecycle solution can scan containers in registries before they're deployed, perform policy checks, and let security teams decide whether to block non-secure or out-of-compliance builds. Issues can be channeled to the appropriate owners using Kubernetes-native environment segregation based on subscription, cluster, and namespace. An API console and command-line interface (CLI) should be provided so that developers can address these risks more quickly and easily.

Deploy

Once a build has been approved, Kubernetes environments, container hosts, and clusters should be hardened using Open Policy Agent (OPA)-based configuration rules and Kubernetes admission policies.

Run

Shift-left doesn't mean ignore-right. Security teams need to continuously assess Kubernetes clusters to identify misconfigurations and threats and suggest remediation steps to mitigate the risks they pose. Real-time scanning can help them identify data access from the container to the cloud, lateral movement between the Kubernetes cluster and the cloud, or the escalation of privileges. To gain a single, prioritized view of risk for an organization's environment, the solution should also provide visibility into toxic combinations showing how different risk factors might combine into an attack path to critical resources.

Effective runtime container security depends on agentless scanning across the full container stack:

- Mapping service accounts and network configurations, and performing analysis into cluster structure and context, network, identities, and secrets. In this way, security teams can gain insight beyond the container to encompass an entire view of the cloud and workloads, including effective network exposure showing effective paths to and from the container, and effective permissions to understand the full scope of the risk associated with each container.
- Scanning images of all running containers using snapshots of container hosts.
- Scanning workloads themselves, including applications, containers, and VMs.
- Pulling containers from the disk volume and scanning their images for vulnerabilities, malware, and exposed secrets as well as detecting technologies.
- Scanning registries where container images are stored

Kubernetes audit logs and reports mapped to the Center for Internet Security (CIS) Foundation Benchmarks for Kubernetes can help you assess the security of your Kubernetes environments and ensure compliance.

Essential function checklist to address key container security use cases

To fully secure your cloud-native environment, make sure you can deploy complete capabilities for scanning, contextual understanding, analysis, and policy enforcement across the CI/CD lifecycle.

Platform

- On-Prem and Cloud
- Managed and Self Managed Kubernetes
- Docker hosts
- Serverless Containers

Code

- Code vulnerability scans
- Secrets scanning
- Exposed secrets scanning
- Host configuration scanning

Build

- Container image vulnerability assessment in CI/CD
- Container registry scanning
- Container security
- Container image signing
- IaC scanning (Terraform, K8s YAML, Helm, Docker, etc.)

Deploy

- Custom OPA-based Kubernetes configuration rules
- Container host hardening
- Kubernetes admission control policies

Run

- Kubernetes security posture management (KPSM) and cluster hardening
- Full lifecycle container image scanning and vulnerability assessment
- Agentless container image scanning using cloud native APIs
- Container effective network exposure assessment
- Container identity entitlement assessment
- Kubernetes network topology analysis
- Kubernetes network traffic analysis
- Kubernetes-to-cloud lateral movement paths
- Kubernetes secrets analysis
- Runtime sensor for the Kubernetes environment
- Kubernetes admission controller
- Cluster threat detection (Kubernetes audit log detection)
- Container runtime threat detection
- Contextual vulnerability prioritization

Explore industry-leading container security/KSPM

Organizations of all sizes, including more than 35 percent of the Fortune 100, rely on Wiz solutions to reduce risk in containerized applications across the development lifecycle. To see the Wiz Operating Platform in action, schedule a live demo with a Wiz expert today.

About Wiz

Wiz secures everything organizations build and run in the cloud. Founded in 2020, Wiz is the fastest-growing software company in the world, scaling from $1M to $100M ARR in 18 months. Wiz enables hundreds of organizations worldwide, including 30 percent of the Fortune 500, to rapidly identify and remove critical risks in cloud environments. Its customers include Salesforce, Slack, Mars, BMW, Avery Dennison, Priceline, Cushman & Wakefield, DocuSign, Plaid, and Agoda, among others. Wiz is backed by Sequoia, Index Ventures, Insight Partners, Salesforce, Blackstone, Advent, Greenoaks and Aglaé. Visit https://www.wiz.io/ for more information.