Red Hat

Whitepaper

Kubernetes-native security

What it is and why it matters

Introduction

In the last 5 years, infrastructure software has benefited from tremendous innovation in cloud-native technologies and architectures. These approaches enable organizations to develop and run applications in ways that are more scalable, flexible, and dynamic than previously possible. Containers, microservices, declarative APIs, and software primitives abstract away underlying compute, storage, and networking resources. Coupled with continuous delivery and DevOps practices, companies of every size across every industry can now release software with greater speed and reliability.

At the core of this systemic shift is the open source container orchestration system Kubernetes. Kubernetes is the de facto standard for automating the deployment and management of cloud-native applications in production.

The adoption of cloud-native technologies creates new security challenges—as well as opportunities to enhance existing security strategies. This cloud-native evolution creates a new threat environment, one whose attack surface is as dynamic, fast-moving, and active as containers themselves. Applications no longer fit within a neat boundary and are intentionally broken up, often into microservices, which means security must operate across interconnected components that interface autonomously. At the same time, key characteristics of containers can be exploited by bad actors to carry out attacks with greater speed and scale, and within shorter time spans, than before.

Similar to virtual machines and public cloud environments, containers and Kubernetes introduce new infrastructure layers, each with their own necessary security measures. In response to these changes, products within the security industry usually take one of three approaches:

1. Host- or network-based solutions extend their protections to containers.
2. Solutions focus on securing container environments.
3. Solutions focus on securing containers and Kubernetes.

Comparing approaches to container security

Key takeaway: Kubernetes-native security goes beyond existing container security approaches to protect Kubernetes environments.

The first category of security tools focuses on protecting cloud-based environments with controls for virtual machines and endpoints (or their networks) with capabilities such as firewalling. The majority of these tools address a specific use case (for example, vulnerability management, intrusion detection, or web application firewalling), and they lose relevance when securing containers since the focus shifts to protecting workloads and their data, not IP addresses or servers.

These options do not provide security across the entire life cycle of cloud-native applications, nor were they designed to deal with the dynamic, highly scalable, and ephemeral nature of containerized applications.

The second category comprises solutions that provide protection across the entire container life cycle and have a container-centric architecture. The majority of container security tools fall into this category. They focus their capabilities only at the level of individual containers, the container runtime or engine (for example, Docker), and the artifacts used to run those containers—namely container images. The downside to this approach is that they do not provide security at the level of Kubernetes itself.

The third category goes beyond container-centric solutions to deliver security that is uniquely architected for Kubernetes environments. This approach, a Kubernetes-native architecture, provides protections across the entire cloud-native application life cycle and addresses threat vectors for both containers and Kubernetes. Red Hat® Advanced Cluster Security for Kubernetes has pioneered this Kubernetes-native approach.

What is Kubernetes-native security?

Key takeaway: A security solution must meet 6 key criteria to be considered Kubernetes-native.

Kubernetes-native security is based on a single principle: security is implemented most effectively when it is aligned with the system that is responsible for managing all of an organization's containerized applications. A security platform must exhibit the following characteristics to be considered Kubernetes-native:

1. Directly integrates with the Kubernetes API server to gain visibility into Kubernetes workloads and infrastructure
2. Assesses vulnerabilities in Kubernetes' software
3. Bases its security functionality, including policy management, on resources within the Kubernetes object model, including deployments, namespaces, services, pods, and others
4. Analyzes declarative data from Kubernetes-specific artifacts (for example, workload manifests) and configurations
5. Uses built-in Kubernetes security features to handle enforcement for greater automation, scalability, and reliability
6. Deploys and runs as a Kubernetes application, including integration and support for common tools in cloud-native toolchains

Organizations seeking to secure their Kubernetes environments should evaluate and prioritize security needs across key use cases that span visibility, vulnerability management, network segmentation, configuration management, compliance, threat detection, and incident response. A Kubernetes-native security platform that meets all 6 criteria can comprehensively address these and other use-case-based security needs.

The benefits of Kubernetes-native security

Key takeaway: A Kubernetes-native security solution delivers several key benefits that other solutions do not: increased protection, reduced time and cost, and minimal operational risk.

Organizations that adopt a Kubernetes-native security approach stand to benefit in 3 crucial ways:

1. **Increased protection:** Kubernetes-native security eliminates blind spots across the orchestrator attack surface, which enables security operators to discover critical vulnerabilities and threat vectors they would otherwise miss.
2. **Reduced time and cost:** A Kubernetes-native approach reduces the overall investment in time, effort, and personnel required to implement security. It also streamlines security analysis, investigation, and remediation by collecting additional context specific to Kubernetes applications and infrastructure.
3. **Minimal operational risk:** A Kubernetes-native security solution minimizes impact to critical operations by enabling enforcement that benefits from the scalability and resiliency of the orchestrator itself. This approach also avoids the conflict and complexity that can result in operational risk to critical applications.

Increase protection with Kubernetes-native security

When organizations shift from running individual containers to orchestrating them with a system like Kubernetes, they gain the operational benefits of being able to run on nearly any infrastructure, as well as consistent service discovery and load balancing, and automated upgrades and rollouts. But at the same time, the infrastructure attack surface expands with the introduction of this new platform. Vulnerabilities can emerge from open source software, unfamiliar configurations, and newly discovered attack vectors.

The developer- and operations-friendly abstractions that Kubernetes makes available can easily create blind spots for traditional security operators. Containers and the applications they make up are no longer fixed to physical boundaries and resources such as servers; because Kubernetes makes compute, networking, and storage fully programmatic, security gaps can persist without appropriate visibility.

Remove blind spots with deeper visibility and insights

Key takeaway: Kubernetes-native security solutions give organizations a comprehensive understanding of their Kubernetes security.

Effective security starts with visibility into a given environment and its overall security state. Kubernetes-native security offers direct visibility into Kubernetes itself, which is achieved by integrating with the Kubernetes API server. This capability provides security monitoring not just for the containers running within Kubernetes clusters but specifically for the resources and objects as Kubernetes sees them. These abstractions—deployments, daemonsets, services, pods, and other resources—reference controllers, network groupings, and workloads. Gaining insight into them provides critical application-level context.

In contrast, container-centric security solutions only maintain visibility at the level of individual, ephemeral containers and their corresponding images, lacking awareness across entire applications. Red Hat Advanced Cluster Security complements activity monitoring within individual containers with deployment-centric visibility that gives users a comprehensive, on-demand picture of Kubernetes applications and their security.

Visibility into configurations

Direct insight into Kubernetes lets organizations assess the security posture of Kubernetes itself by evaluating configurations and settings. Kubernetes-native security automates policy checks to enforce security best practices and identify misconfigurations that can increase the risk of exposing application data. Red Hat Advanced Cluster Security continuously assesses hundreds of Kubernetes configurations to quickly determine the overall security of Kubernetes environments. These configurations include, but are not limited to:

- Role permissions.
- Access to secrets.
- Allowed network traffic.
- Settings for control plane components.
- Other settings.

By comparison, container-centric solutions focus on configurations of the container runtime engine and individual containers (such as container privileges and kernel capabilities). Because they may miss categories of Kubernetes controls, unseen insecure configurations can give rise to exploits.

Visibility into compliance

Another parallel need for Kubernetes-native visibility is compliance, including industry standards such as PCI, HIPAA, SOC 2, and FedRAMP. Even if an organization's security posture is strong, the security of Kubernetes applications and infrastructure must be appropriately accounted for to determine compliance. Audits and certifications may require heightened control measures that include comprehensive visibility into Kubernetes.

The Kubernetes-native approach from Red Hat Advanced Cluster Security applies checks specific to Kubernetes clusters that are automatically mapped to existing compliance controls, instantly giving organizations an understanding of their pass and fail rates. For example, restricting network access between the internet and services processing sensitive data or discovering vulnerabilities in the version of Kubernetes being used.

Container-centric security focuses on compliance checks agnostic to orchestration system, typically based on the Center for Internet Security's (CIS) Benchmark for Docker, resulting in an incomplete assessment of a given environment's overall compliance.

Visibility into workload isolation

Containers inherently provide less isolation than virtual machines, especially in scenarios that require multi-tenancy. One of the key benefits of Kubernetes is the number of options available to provide isolation between workloads when running containers. Isolation is a fundamental starting point

for securing any Kubernetes environment, and these options include physical boundaries (clusters, nodes) as well as virtual boundaries. Both can be implemented using features such as namespaces and network policies.

Kubernetes-native security helps teams understand and implement workloads isolated within Kubernetes using these different options.

Container-centric security largely ignores these different boundaries by focusing on the hosts where different containers run, at the expense of knowing whether applications and their services are reasonably segregated.

Red Hat Advanced Cluster Security provides insight that:

- Is based on an understanding of clusters, namespaces, and pods.
- Collects data from every single host.
- Integrates with Kubernetes network policies to provide a full understanding of how applications are isolated.

An effective security strategy is only as strong as its foundation, and comprehensive visibility into Kubernetes is essential to successfully protecting cloud-native applications.

Discover critical vulnerabilities and threat vectors

Key takeaway: Kubernetes-native security solutions discover and protect against Kubernetes vulnerabilities, attack vectors, and misconfigurations.

A key aspect of securing workloads running on Kubernetes is to understand and protect against threats that are specific to cloud-native environments—threats not just to containers but also to the Kubernetes platform itself. The breadth and complexity of Kubernetes creates an expansive set of attack vectors that continue to evolve as the Cloud Native Computing Foundation (CNCF), open source communities, and contributing organizations continue to advance the platform. Kubernetes-specific security threats with public impact include:

- A Kubernetes dashboard that was configured insecurely in Tesla's cloud environment,[1] allowing attackers to gain access to account credentials and mine cryptocurrency.
- A service at Shopify that was vulnerable to a server-side request forgery (SSRF)[2] that could have allowed an attacker to obtain kubelet credentials and replay them to gain root access to all containers.

The cloud-native community also recognizes these unique threats to Kubernetes. An inaugural 2020 security audit[3] determined areas of security weakness within Kubernetes, including a threat model that spans 8 major Kubernetes components. Kubernetes-native security in Red Hat Advanced Cluster Security builds on this and other research efforts across the cloud-native ecosystem to deliver unique capabilities to identify vulnerabilities, misconfigurations, and exploits across Kubernetes.

---

1 *Newman, Lily Hay. "Hack Brief: Hackers Enlisted Tesla's Public Cloud to Mine Cryptocurrency." Wired, 28 Feb. 2018.*

2 *Kerner, Sean Michael. "How Shopify Avoided a Data Breach, Thanks to a Bug Bounty." eWeek, 17 Dec. 2018.*

3 *"Kubernetes Final Report." GitHub (kubernetes/community/wg-security-audit/findings), 6 Aug. 2019.*

Detect Kubernetes-specific vulnerabilities

Among the many components that comprise Kubernetes, the API server is arguably the most critical one that must be guarded, given its principal role in orchestrating and managing containerized applications. One example of a critical vulnerability that impacts the Kubernetes API server is CVE-2018- 1002100, which would allow a bad actor to compromise an entire cluster while making authorized requests, making it even more difficult to identify. Until its discovery more than 3 years after the initial release of Kubernetes, the vulnerability had existed in every version.

Kubernetes-native security automatically discovers known Kubernetes vulnerabilities, so cluster administrators and operators can address some of the largest risks to their environments.

In contrast, container-centric solutions focus only on vulnerabilities in container images. And while containers with fewer vulnerabilities may be more secure, vulnerable Kubernetes infrastructure could leave them susceptible to compromise. Red Hat Advanced Cluster Security proactively identifies clusters with the most Kubernetes vulnerabilities and allows users to specify policies based on individual vulnerabilities.

Detect Kubernetes-specific misconfigurations

In addition to vulnerabilities, misconfigurations of core Kubernetes controls can give rise to exploits and exposure of sensitive application data, especially when the misconfiguration can be automatically propagated across clusters. These misconfigurations can stem from either user error or lack of familiarity with Kubernetes. Examples of key misconfigurations include giving an unqualified user or service account full cluster administrative permissions using role-based access controls (RBAC), or unnecessarily exposing Kubernetes secrets by giving deployments the ability to pull secrets even when they are not required.

Red Hat Advanced Cluster Security integrates with these controls to quickly alert organizations when misconfigurations occur, helping them to achieve better configuration management overall.

Container-centric solutions focus on problematic configurations of the container runtime or individual containers, resulting in incomplete protections that can be circumvented. Even if a misconfiguration for a given container is addressed, a misconfiguration in Kubernetes could allow an attacker to compromise the node where that container is running. In this example, the container is still compromised, even though the container itself was configured correctly.

Red Hat Advanced Cluster Security addresses this shortcoming by looking for misconfigurations in both Kubernetes and containers.

Protect ingress and egress network communications

Kubernetes also incorporates extensible networking options, including how pods communicate with each other using network policy specifications. However, these settings have security implications that can be easily overlooked. For example, if no Kubernetes network policies are applied to a given pod, then all network traffic (ingress and egress) is allowed. By default, Kubernetes applies no network policies.

The Kubernetes-native security approach instantly understands network topology based on how network policies are configured, often by DevOps and operations teams.

Container-centric solutions understand how network traffic is configured based on rules within their own proprietary interfaces, outside of Kubernetes. Red Hat Advanced Cluster Security ensures consistent, open, standardized network traffic rules are maintained within Kubernetes itself.

Successfully protecting containerized applications must start with securing the container infrastructure they run on. This step requires understanding critical vulnerabilities, threat vectors, and misconfigurations specific to Kubernetes, which can be achieved only with a Kubernetes-native security solution.

Reduce time and cost with Kubernetes-native security

One of the most significant advantages of Kubernetes is that it provides a unified platform for provisioning and operating infrastructure services, whether in the cloud or on-premise. This eliminates operational complexity and inconsistency and streamlines workflows across developers, operations, and security teams to save time and reduce cost. Organizations should look to secure their Kubernetes environments in a similar manner, and choose solutions that reduce the effort required.

Key questions you should ask when evaluating security solutions include:

- Does the solution shorten the learning curve for security and DevOps teams?
- Can the solution help teams accelerate incident analysis, investigation, and remediation?
- Does the solution provide automation and relevant data collection?

A Kubernetes-native architecture satisfies these needs in several critical ways.

An easier learning curve for teams

Key takeaway: Kubernetes-native security solutions simplify Kubernetes security by integrating with open, standard abstractions and tooling already used by DevOps teams.

Complexity is the enemy of security. Kubernetes, with its multiple components and associated tooling, requires teams to learn new skills and adopt new workflows across development and operations. The last thing they need is additional complexity from a security solution that is completely separate from these new investments in toolchains and processes.

Use a single, familiar framework

A Kubernetes-native architecture minimizes the number of new interfaces, configurations, and resource models that users must learn to secure their cloud-native workloads. To build and ship containerized applications, developers and operators already use standardized abstractions such as Kubernetes manifests. Security functions in Red Hat Advanced Cluster Security are based on these same abstractions, so development, operations, and security teams use a single framework to build, ship, and run applications.

For example, Red Hat Advanced Cluster Security users can view the metadata of Kubernetes manifest files to discover valuable security-relevant context for an entire application and its intended function. Container-centric solutions, by comparison, focus on Dockerfiles, which only include the commands that can be called to build an image.

Red Hat Advanced Cluster Security also takes advantage of standardized package management tools, deploying them the same way as other containerized applications. For example, Red Hat Advanced Cluster Security is deployed using Helm charts, whereas container-centric solutions may

rely on package management options that are not necessarily Kubernetes-friendly. Basing security on open, standard abstractions and packaging helps eliminate misalignment between security teams and other stakeholders, ultimately streamlining collaboration and saving valuable time.

Leverage *configure once, use everywhere* model to easily *shift left*

Kubernetes-native security also reduces the effort required to implement secure configurations by using Kubernetes to orchestrate declarative configurations across the environment in a *configure once, use everywhere* model. Using this approach, organizations can easily automate hundreds of checks for security best practices and policies.

With Red Hat Advanced Cluster Security, users can propagate a single configuration, such as a network policy, across all pods in a deployment rather than configuring system-level controls on every host in a cluster, as is required by some container-centric solutions. Container-centric solutions do not take full advantage of the scalability and resiliency that is inherent in Kubernetes; Red Hat Advanced Cluster Security applies these attributes of native Kubernetes controls to an organization's advantage.

The security of containerized applications is impacted by controls implemented along the software supply chain, before those containers are actually running. This approach presents an opportunity for developers and DevOps teams to use their existing knowledge of Kubernetes to help an organization *shift left* and adopt a mindset of security as code. This is a concept many teams are familiar with, as it is similar to their successful infrastructure as code methodologies today.

Kubernetes-native security complements these existing initiatives by integrating functionality such as deploy-time policy enforcement using the dynamic admission control built into Kubernetes. Container-centric solutions use proprietary architectures to enforce policies at the time of deployment, missing the opportunity for developers and DevOps to interweave security more deeply with Kubernetes-related toolchains.

By aligning security with the abstractions and toolchains developers and DevOps teams are already familiar with, Red Hat Advanced Cluster Security helps organizations save time and reduce cost when executing cloud-native security strategies.

Faster analysis and remediation

Key takeaway: Kubernetes-native security solutions streamline investigation, incident response, and risk assessment by capturing rich context directly from Kubernetes.

Today's security teams have to do too much work to get the answers they need. They operate a complex patchwork of tools, relying on manual workflows and custom integrations to tie them together. This often means sifting through an endless stream of alerts; containers can make their job even harder. Security operators face substantial challenges protecting cloud-native environments, including:

- Incidents and their new, nonlinear threat patterns (indicators of attack and compromise) are scattered across application components and distributed environments, because containers are dynamically orchestrated across machines.
- Existing tools and workflows cannot keep up with the high volumes of data produced by large numbers of containers.

- Many traditional approaches to incident response, which rely on the preservation of stateful data in order to study an attack and learn the attacker's techniques and craft, are ineffective because of the ephemeral nature of containers.

Detect threats more accurately

The detailed context Kubernetes-native security provides can speed investigations of security incidents and the subsequent remediation of their underlying issues. Containerized applications, especially those using microservices architectures, allow higher fidelity threat detection because it's easier to observe and identify anomalies in smaller components with predictable activity. Combining this knowledge with additional context from Kubernetes makes it far easier to detect actual threats versus false positives.

Context from Kubernetes includes deployment-wide information about:

- What process will execute.
- If any resource limits exist to prevent containers from impacting their neighbors.
- Which privileges and capabilities are granted to individual containers.
- If the container root file system will be writable.
- What block devices, configurations, or secrets are present.

It also includes valuable metadata: Labels help establish what an application is, while annotations add descriptions about the application.

Red Hat Advanced Cluster Security uses this context to align relevant data across workloads—information about container images and system-level activity from individual containers at runtime—with configuration data from Kubernetes. The security of Kubernetes impacts the security of individual containers and vice versa.

By capturing data about individual containers and how they relate to each other, Red Hat Advanced Cluster Security hones in on security issues distributed across large-scale environments. Container-centric solutions miss capturing this Kubernetes metadata, leaving security operators with only a partial understanding of the incident's root cause.

A Kubernetes-native solution integrates disparate data across containers, Kubernetes, and the application life cycle from the time images are built to when containers run. With this information, teams are able to analyze how runtime activity compares to what containers were declaratively configured to do based on Kubernetes settings. As a result, Red Hat Advanced Cluster Security can more accurately detect anomalous and suspicious runtime activity, while avoiding false positives and reducing alert fatigue.

Container-centric solutions, in contrast, base their threat detection on observed information from running containers only, which results in a stream of alerts that security operators must handle. By automatically collating threat-related information across the life cycle as well as across containers and Kubernetes, Red Hat Advanced Cluster Security eliminates silos of information that security operators have to piece together manually.

Implement risk-based security prioritization

Kubernetes-native security also correlates information about vulnerabilities with an understanding of configurations and observations of runtime activity to assess the likelihood that particular vulnerabilities can be exploited.

For example, exploiting a vulnerability that requires certain privileges depends on whether those privileges exist for containers in a given deployment. Similarly, a vulnerability exploited by writing to a container's filesystem may not be worrisome if the filesystem is read-only. And a high level of network access might be configured for a certain deployment, because it is in a nonproduction lab cluster or an external web application.

Red Hat Advanced Cluster Security determines a level of risk associated with each deployment within the environment and ranks them for security operators. Container-centric solutions primarily assess risk based on image vulnerabilities and standardized measures like the Common Vulnerability Scoring System (CVSS), making any type of risk scoring ineffective. A holistic understanding of risk is achievable only with full insight into Kubernetes.

Incident analysis and response is only as effective as the data a security solution captures and presents for investigation. With finite time and resources, security operators need the faster analysis and remediation of Kubernetes-native security.

Minimize operational risk with Kubernetes-native security

Kubernetes is a robust, scalable platform that runs nearly any application, including critical workloads. Kubernetes development focuses on features that bring operational resilience, including:

- Self healing, including the ability to handle container restarts.
- Rescheduling.
- Killing containers when they fail health checks.

Kubernetes is also highly scalable, applying the same principles that Google uses to run billions of containers every week. Kubernetes infrastructure meets demanding operational requirements for availability (uptime), reliability (mean time between failures), latency, and other critical metrics with direct business impact. And Kubernetes does this without increasing operational risk to applications, infrastructure, or the business.

Some security tools provide active security enforcement that can directly impact an environment by killing containers or processes (or preventing them from launching), blocking network traffic, or restricting system calls that an application can make. These tools increase the risk to the business for 2 primary reasons:

1. Actions taken in error can result in an application outage.
2. Tools become additional dependencies that operations teams must worry about. If they fail closed they can significantly disrupt operations, and if they fail open, the organization is unprotected.

A security tool must mitigate against these risks.

Scalable enforcement

Key takeaway: Kubernetes-native security enables highly scalable security enforcement, while minimizing potential disruption to critical applications, using native Kubernetes security controls.

Using the orchestrator to carry out security enforcement functions minimizes the operational risk and provides greater scalability in Kubernetes environments. This approach was not possible when infrastructure platforms lacked native security capabilities. Today, Kubernetes includes:

- Network policies for network segmentation.
- Admission controllers for intercepting requests to the Kubernetes API server.
- Secrets for storing sensitive credentials.
- Role-based access control for granting authorization to users and services accounts.
- Many other capabilities.

This architectural approach eliminates additional security tooling for certain enforcement functions.

In contrast, container-centric solutions pose several operational risks to an organization. For example, container-centric security solutions often have to use an in-line proxy to restrict network traffic between containers. Or they may take more intrusive measures, such as overwriting IP rules for firewalling.

Similarly, to prevent containers from deploying with critical vulnerabilities or when they contain packages that are not allowed, a container-centric solution might rely on a shim to intercept and override requests to the container engine. This introduces single points of failure throughout an environment—if an in-line proxy or container engine shim fails, then applications may also fail.

Red Hat Advanced Cluster Security stores policies for network segmentation, admission control, and other security functions in Kubernetes itself, which removes operational dependencies. If Red Hat Advanced Cluster Security becomes unavailable for some reason, security policies continue to be enforced. The same cannot be said for container-centric solutions.

Container-centric solutions also introduce scalability problems because, in some cases, separate enforcement components must run on every host within a cluster. This approach incurs performance overhead, impacts resource availability for application containers, and requires additional attention from operations teams, increasing an organization's overall operational burden.

With Red Hat Advanced Cluster Security, the orchestrator is the key security enforcement point, using the inherent capabilities of Kubernetes to deliver highly scalable and reliable enforcement. Cluster administrators and infrastructure teams can focus on Kubernetes operations without the distraction of third-party components. And organizations gain new security controls with the robustness and scale of the Kubernetes system, backed by an ecosystem of hundreds of companies and a community of thousands of developers.

A Kubernetes-native security solution is also agnostic to community-driven architectures. One example is the Container Network Interface (CNI), a standard networking interface specification. Red Hat Advanced Cluster Security's network segmentation capabilities operate consistently with any CNI plugin.

Container-centric solutions may be dependent on or incompatible with certain plugins, complicating the operational issues an organization must address. The Red Hat Advanced Cluster Security controls do not have dependencies on specific tools or plugins, allowing DevOps and operations teams to use the ones that meet their needs.

Eliminate operational conflict

Key takeaway: Kubernetes-native security helps reduce operational issues that stem from inconsistent configuration, lack of coordination, and user error.

To minimize operational risk, organizations must also avoid inconsistent security configurations that can lead to unpredictable application behavior and complexity that contributes to user errors. A Kubernetes-native architecture reduces the likelihood of these types of operational issues.

Security and DevOps teams that implement different tools, controls, and policies can create inconsistency in their security configurations. Inconsistency can also arise from conflict between configurations at different layers of the infrastructure stack. For example, settings may be configured at the container level that conflict with those specified at the Kubernetes level.

Kubernetes-native security allows development, operations, and security teams to view and manage policies and enforcement in a centralized way within Kubernetes itself. With Red Hat Advanced Cluster Security, network segmentation policies and admission control policies persist in Kubernetes for easy reference.

Container-centric solutions force users to implement controls using different tools, resulting in a lack of organizational coordination. For example, a DevOps user may be using Kubernetes network policies to restrict allowed network traffic to an application, while a security user would use a proprietary firewall. This can result in conflicting configurations and gaps in protection.

Similarly, inconsistencies in Kubernetes- and container-level configurations can unexpectedly impact running applications. For example, container-centric solutions can restrict what operating system calls a particular container can make, even if no similar restrictions are configured within Kubernetes using pod security policies. As a result, someone looking at Kubernetes may not understand what is causing an operational issue because the relevant configuration is specified at a different layer.

And in cases where restrictions conflict, traffic may not be allowed or restricted as intended. For example, a Kubernetes network policy allows traffic between container A and container B, but a container-centric in-line proxy restricts traffic between those containers. The result is operational confusion.

Kubernetes-native security treats Kubernetes as a source of truth for security functions and lessens the amount of context switching required from operations and site reliability engineering (SRE) teams. Security issues are immediately mapped to the Kubernetes objects and resources—things these teams already work with daily to ensure the availability and reliability of Kubernetes itself.

This approach reduces the operational complexity that contributes to user errors that negatively impact Kubernetes environments. Container-centric solutions require operations and SRE personnel to manually correlate security issues between vendor-specific information and the data within Kubernetes. Red Hat Advanced Cluster Security enables users to view its information in a format that is already aligned with what Kubernetes presents, making it faster and simpler for cluster operators to pinpoint issues and solve them using chosen toolchains and workflows.

Conclusion

Cloud-native technologies have been transforming and will continue to transform how organizations run their applications and, consequently, how they must think about security. They enable the adoption of DevSecOps and other collaborative practices, and move our industry toward a model of security as code. At the center of this generational shift in infrastructure software is Kubernetes, the leading container orchestration system. And although there are many approaches to protecting container environments, Kubernetes-native security provides deeper insight, faster analysis, and simpler operations.

Red Hat's Kubernetes-native security delivers several unique benefits to organizations building and running containerized applications:

- Provides increased protection by eliminating blind spots and discovering critical vulnerabilities and misconfigurations unique to Kubernetes.
- Saves time and reduces cost by shortening the learning curve for teams and accelerating investigation and remediation with valuable context from Kubernetes.
- Minimizes overall operational risk by using Kubernetes for scalable enforcement functions and eliminating operational complexity that arises from inconsistent configurations and user error.
- Provides a standardized platform for security use cases across the cloud-native application life cycle that can be used by development, operations, and security teams.

With Red Hat Advanced Cluster Security, organizations can more effectively secure their Kubernetes environments anywhere, in production, at scale.

North America
1 888 REDHAT1
www.redhat.com

Europe, Middle East, and Africa
00800 7334 2835
europe@redhat.com

Asia Pacific
+65 6490 4200
apac@redhat.com

Latin America
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com
#F28129_0421