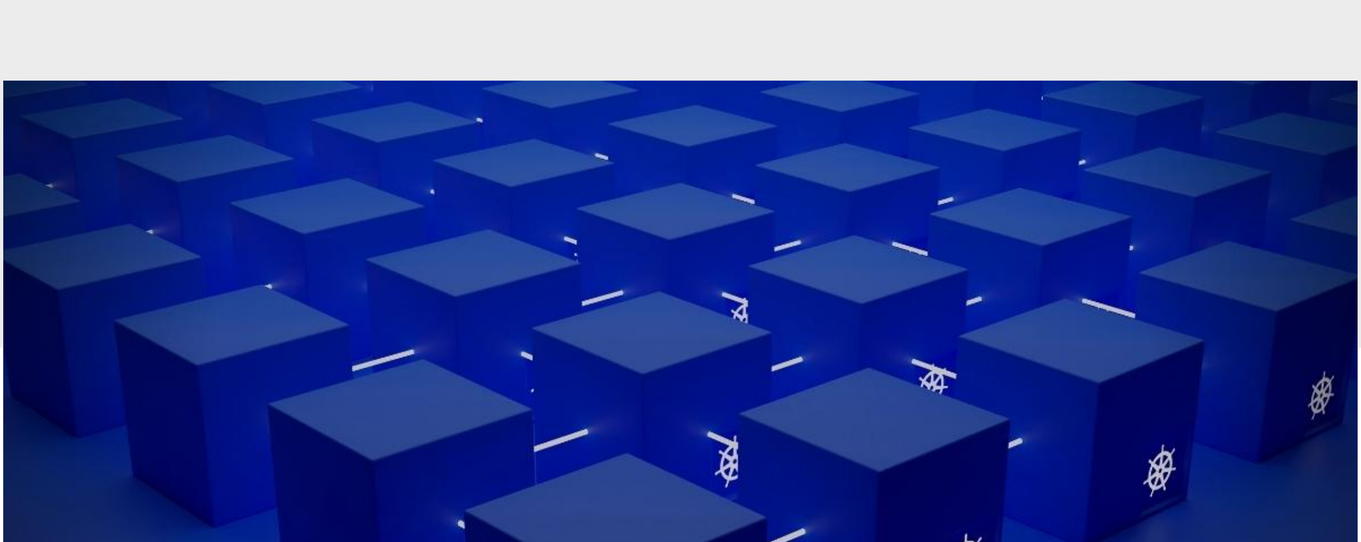


Mastering Kubernetes Security | Top Strategies Recommended by OWASP

April 6, 2023
by SentinelOne

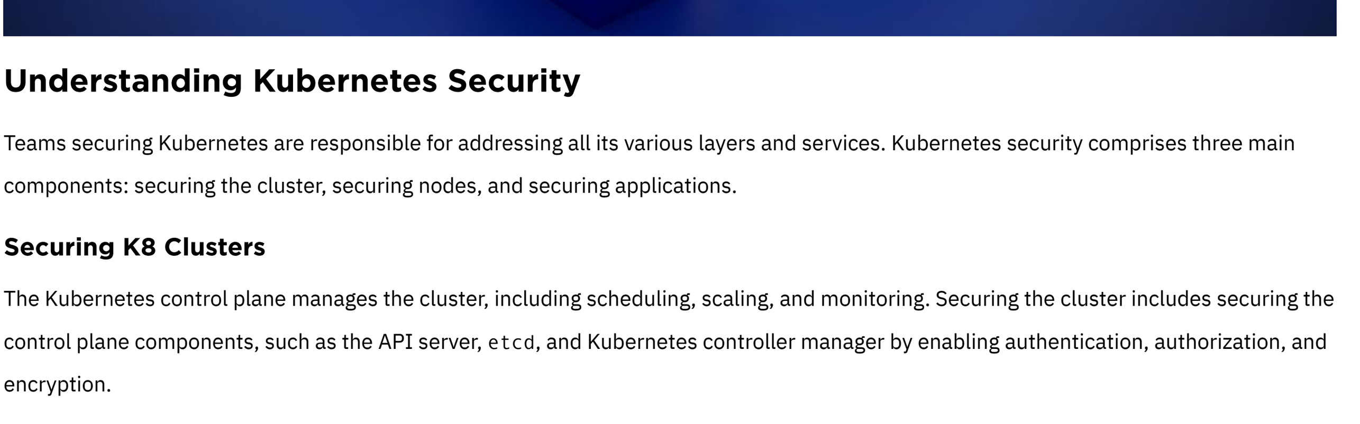
Kubernetes, a popular open-source container orchestration system, has gained popularity among enterprises for its ability to manage and automate large-scale containerized workloads. However, as with any technology, inherent security risks must be considered and addressed.

In this post, we explore the top ten Kubernetes security risks and provide recommendations for mitigating these risks.

What is Kubernetes?

Kubernetes, commonly referred to as "K8s", is a container orchestration system that automates the deployment, scaling, and management of containerized workloads. It was originally developed by Google and is now maintained by the Cloud Native Computing Foundation (CNCF).

Kubernetes is a powerful tool that offers self-healing, auto-scaling, and service discovery features. In addition, it allows developers to deploy their applications as workloads that can run on any platform that supports Docker containers.



Understanding Kubernetes Security

Teams securing Kubernetes are responsible for addressing all its various layers and services. Kubernetes security comprises three main components: securing the cluster, securing nodes, and securing applications.

Securing K8 Clusters

The Kubernetes control plane manages the cluster, including scheduling, scaling, and monitoring. Securing the cluster includes securing the control plane components, such as the API server, `etcd`, and Kubernetes controller manager by enabling authentication, authorization, and encryption.

Securing K8 Nodes

Nodes are the worker machines in a Kubernetes cluster that run the containers. Nodes can be secured through the host operating system, by configuring network security, and by securing the Kubernetes runtime environment. Removing unnecessary user accounts and ensuring that nothing runs as root are all best practices to consider when securing K8 nodes.

Securing K8 Applications

In Kubernetes, a pod is a container used to run an application. Securing these applications means securing the pod. Kubernetes provides several security features to help secure applications. These features can be used to limit resource access, enforce network policies, and enable secure communication between containers.

Top 10 OWASP Kubernetes Security Risks & Recommendations

The OWASP Foundation was created to improve software security through community-led, open-source software projects. Here are the top ten strategies recommended by OWASP for securing Kubernetes ecosystems.

1. Insecure Workload Configurations

Kubernetes manifests contain a plethora of configurations that can affect the reliability, security, and scalability of a given workload. These configurations should be audited and remediated continuously to prevent misconfigurations. However, some high-impact manifest configurations are more likely to be misconfigured than others. Some examples are:

- Running application processes as root inside a container is a common misconfiguration in many clusters. Though root may be required for some workloads, it should be avoided when possible. If the container were to be compromised, the attacker would have root-level privileges, allowing actions such as starting a malicious process that otherwise wouldn't be permitted with other users on the system.
- To limit the impact of a compromised container on a Kubernetes node, it is recommended to utilize read-only filesystems when possible. This prevents a malicious process or application from writing back to the host system. Read-only filesystems are a key component to preventing container breakout.
- Privileged containers should be disallowed as they can access additional resources and kernel capabilities of the host. Workloads running as root combined with privileged containers can be devastating as the user can access the host completely. This is, however, limited when running as a non-root user. Privileged containers are dangerous as they remove many of the built-in container isolation mechanisms entirely.

While many security configurations are often set in the `securityContext` of the manifest itself, other misconfigurations can be detected elsewhere. They must first be detected in both runtime and code to prevent misconfigurations. It is imperative to enforce that applications run as non-root users, run in non-privileged mode, and set 'AllowPrivilegeEscalation' to 'False' to disallow child processes from gaining more privileges than their parents.

Security teams can use tools such as Open Policy Agent as a policy engine to detect common misconfigurations like the ones listed above. Using the CIS Benchmark for Kubernetes is a good starting point for discovering misconfigurations. However, it is important to continuously monitor and remediate any potential misconfigurations to ensure the security and reliability of a Kubernetes workload.

2. Supply Chain Vulnerabilities

At various phases of the development lifecycle supply chain, containers take on many forms and each will present its own unique security challenge. This is because a single container may rely on hundreds of external, third-party components, diluting the level of trust at each phase. The most common supply chain vulnerabilities are below:

- Image integrity – Container images are made up of layers, each bringing possible security risks. Since container images use third-party packages extensively, they can be dangerous to run within a trusted environment. To mitigate this, it's important to ensure image integrity by validating software at each phase using in-toto attestations. Doing so increases the SLSA level of the build pipeline, which means it is more resilient against attacks. Additionally, using image signing and verification through cryptographic key-pairs can detect tampering with the artifacts throughout the DevOps workflow, which is an essential step in building a secure supply chain.
- Image composition – Container images contain layers, each presenting different security implications. A properly constructed container image reduces the attack surface and increases deployment efficiency. Thus, it's important to create container images using minimal OS packages and dependencies to reduce the attack surface, considering using alternative base images such as Distroless or Scratch to improve security posture and reduce image size. Furthermore, tools like Docker Slim are available to optimize image footprint for performance and security reasons.
- Known software vulnerabilities – Security flaws are widespread due to the extensive use of third-party packages in container images. Image vulnerability scanning is crucial for enumerating known security issues in container images, which should be used as a first line of defense. Open-source tools such as Clair and trivy statically analyze container images for known vulnerabilities such as CVEs, and should be used as early in the development cycle as reasonably possible.

Enforcing policies to prevent unapproved images from being used is also essential. Kubernetes admission controls and policy engines such as Open Policy Agent and Kyverno can reject workload images that haven't been scanned for vulnerabilities, use a base image that's not explicitly allowed, don't include an approved SBOM, or originate from untrusted registries.

3. Overly-Permissive RBAC

Role-based access control (RBAC) allows the definition of who has access to what resources in a cluster and what they can do with those resources. When configured correctly, RBAC helps prevent unauthorized access and protect sensitive data.

However, if RBAC is not configured correctly, it can lead to overly-permissive settings that allow users to access resources that they should not have access to or perform actions that they should not be able to perform. This can create serious security risks, including data breaches, data loss, and compromise.

Examples of overly-permissive RBAC include the unnecessary use of cluster-admin in Kubernetes. Granting access to this "superuser" role gives unfettered control over every resource in the cluster, which is especially dangerous when used in a ClusterRoleBinding, which grants the all-powerful cluster-admin privilege to every single Pod in the default namespace, making the entire cluster vulnerable to attack.

To prevent such an attack, it is crucial to continuously analyze RBAC configurations and enforce the principle of least privilege (PoLP). This can be achieved by reducing direct cluster access by end users, avoiding using Service Account Tokens outside of the cluster, and auditing RBAC included with installed third-party components. Moreover, deploying centralized policies to detect and block risky RBAC permissions, utilizing RoleBindings to limit the scope of permissions to particular namespaces, and following the official RBAC Good Practices is highly recommended.

4. Policy Enforcement

Policy enforcement involves the implementation of rules and regulations to ensure compliance with organizational policies. In the context of Kubernetes, policy enforcement ensures that the Kubernetes cluster adheres to the security policies set by the organization. These policies could be related to access control, resource allocation, network security, or any other aspect of the Kubernetes cluster.

Policy enforcement is essential for ensuring the security and compliance of the Kubernetes cluster. Failure to enforce policies can lead to security breaches, data loss, and other potential risks. Additionally, policy enforcement helps maintain the integrity and stability of the Kubernetes cluster, ensuring that resources are allocated effectively and efficiently.

Best Practices for Policy Enforcement in Kubernetes

It is essential to follow best practices to ensure effective policy enforcement in Kubernetes. Some of these include:

- Defining policies that align with organizational goals and regulatory requirements.
- Implementing policies using Kubernetes-native resources or policy controllers.
- Regularly reviewing and updating policies to ensure they remain relevant and effective.
- Monitoring policy violations and remediating them promptly.
- Educating users on Kubernetes policies and their importance.

5. Inadequate Logging

Logging is an essential component of any system that runs applications. It involves collecting and storing data about the system's behavior and its applications. Logging in Kubernetes is no different. Kubernetes logs are records of events that occur within a Kubernetes cluster.

These logs can help identify system issues and provide valuable insight into system performance, security breaches, and data loss. Various sources, including application code, Kubernetes components, and system-level processes, can generate Kubernetes logs.

Best Practices for Kubernetes Logging

- Use a Centralized Logging System – A centralized logging system collects and stores logs from all Kubernetes components and applications in a single location. This makes it easier to identify and respond to issues with the system. There are many different centralized logging systems available for Kubernetes. Some popular options include Elasticsearch, Fluentd, and Logstash.
- Use Standardized Logging Formats – Standardized logging formats make searching and analyzing logs from multiple sources easier. This is especially important when using a centralized logging system. Several standardized logging formats are available for Kubernetes, including JSON and syslog. Security teams will need to choose a format supported by their logging system and configure their Kubernetes components and applications to use that format.
- Maintain Complete Logs – When it comes to Kubernetes logging, it's important to log everything: application logs, Kubernetes component logs, and system-level logs. Logging everything ensures a complete picture of system behavior. However, logging everything can also generate a large amount of data. To manage this data, consider setting up log rotation and retention policies. Log rotation policies ensure that logs are rotated and compressed regularly to conserve disk space. Retention policies determine how long logs are kept before being deleted.
- Use Labels and Annotations – Labels and annotations are a powerful feature of Kubernetes that can provide additional context to logs. Labels are key-value pairs that can be attached to Kubernetes objects, such as pods and services. Annotations are similar to labels but can contain larger amounts of information. Labels and annotations can filter and search logs based on specific criteria. For example, security teams can add a label to all pods that are part of a particular application and then search for logs from those pods based on the label.
- Monitor Kubernetes Logs – Monitoring logs regularly allows security teams to identify issues with the system and respond to them quickly. There are many different tools available for monitoring Kubernetes logs. Some popular options include Prometheus, Grafana, and Kibana. These tools support log visualization where alerts can be set up based on specific criteria.
- Set Up Auditing – Setting up auditing in Kubernetes enables teams to track changes to the Kubernetes API server and other Kubernetes components, help identify unauthorized system changes, and ensure compliance with security policies. To set up auditing in Kubernetes, configure the Kubernetes API server to log audit events. These events can then be sent to a centralized logging system for analysis.

6. Broken Authentication

Broken authentication is a vulnerability that allows attackers to bypass authentication and gain unauthorized access to an application or system. Authentication is verifying a user's or system's identity, usually by requiring a username and password. If an attacker can bypass the authentication process, they can gain access to sensitive data, systems, or applications. In Kubernetes, broken authentication can occur due to several factors, including:

- Weak or compromised authentication credentials – If an attacker can obtain a user's credentials, they can bypass authentication and gain unauthorized access to the Kubernetes cluster.
- Misconfigured authentication settings – Kubernetes supports several authentication mechanisms, including X.509 certificates, static tokens, and OAuth tokens. Misconfigured authentication settings can leave the Kubernetes cluster vulnerable to attack.
- Insecure communication channels – Kubernetes uses various communication channels, including the Kubernetes API server, kubelet, and, etcd. An attacker can intercept and manipulate traffic to bypass authentication if these communication channels are insecure.

How to Prevent Broken Authentication in Kubernetes

Preventing broken authentication in Kubernetes requires implementing several security measures, including:

- Strong authentication credentials – Users must use strong and unique passwords or authentication tokens that are not easily guessable.
- Secure communication channels – Communication channels between Kubernetes components must be encrypted using SSL/TLS.
- Proper authentication configuration – The authentication mechanisms used in Kubernetes must be correctly configured to prevent unauthorized access.
- Implementing RBAC – Role-Based Access Control (RBAC) should be used to restrict access to Kubernetes resources based on a user's role.

7. Network Segmentation

Network segmentation divides a network into smaller subnetworks, each isolated. This is done to improve security by limiting the scope of potential attacks. By isolating different parts of the network from each other, network segmentation makes it harder for attackers to move laterally within the network and gain access to sensitive resources.

By default, any workload can communicate with another workload when no additional controls are put in place in a Kubernetes network. An attacker can leverage this default behavior by exploiting a running workload to probe the internal network, move to other running containers, or even invoke private APIs.

How to Implement Network Segmentation in Kubernetes Clusters

Isolating traffic within the context of a Kubernetes minimizes damage and loss should a container become compromised. Several techniques can be used to implement network segmentation in Kubernetes clusters to stop lateral movement and still allow valid traffic to route as normal. Two important techniques are:

- Using Network Policies – Kubernetes supports network policies, which define how traffic flows between pods and namespaces. Using network policies controls which pods can communicate with each other and which ones are isolated from the rest of the cluster.
- Using Network Segmentation Tools – Many third-party tools can implement network segmentation in Kubernetes clusters. The most popular ones include Calico, Weave Net, and Cilium. These tools provide advanced network segmentation capabilities, such as encryption, firewalling, and intrusion detection.

8. Secrets Management

A "secret" is an object in Kubernetes that contains sensitive data such as passwords, certificates, and API keys. Secrets store confidential data that should be inaccessible to other users and processes within the cluster. Kubernetes secrets are stored in `etcd`, a distributed key-value store used by Kubernetes to store all cluster data.

Kubernetes Secrets Management Best Practices

Though secrets are a very useful function in the Kubernetes ecosystem, they need to be handled with caution. Managing Kubernetes secrets can be broken down into the following steps:

1. Deploy encryption at rest – A potential attacker can gain considerable visibility into the state of a cluster by accessing the `etcd` database, which contains any information accessible via the Kubernetes API. Kubernetes offers encryption at rest; a feature introduced in version 1.7 and v1 beta since 1.13. Encryption at rest safeguards secret resources in `etcd`, ensuring that the content of those secrets remains hidden from parties that access `etcd` backups. This feature is still in the beta stage, but it provides an extra layer of security in situations where backups are not encrypted, an attacker has read access to `etcd`.
2. Address security misconfigurations such as vulnerabilities, image security, and policy enforcement – RBAC configuration should also be locked down, and all Service Account and end-user access should be restricted to the least privilege, especially when accessing secrets. Auditing the RBAC configuration of third-party plugins and software installed in the cluster is also necessary to ensure that access to Kubernetes secrets is not granted unnecessarily.
3. Ensure that logging and auditing are in place – This helps detect malicious or abnormal behavior, including access to secrets. Kubernetes clusters produce useful metrics around activities that can be leveraged to detect such behaviors. Therefore, enabling and configuring Kubernetes audit records and centralizing their storage is advisable.

A few more additional tips and tricks include rotating secrets regularly to reduce the risk of secrets being compromised, auditing secret access to detect any unauthorized access to secrets, and using third-party secret management tools such as HashiCorp Vault or CyberArk Conjur to manage Kubernetes secrets.

9. Misconfigured Cluster Components

Kubernetes clusters are composed of various different components from key-value storage within `etcd`, the `kubelet`, the kube-apiserver, and more. All of the components are each highly configurable, meaning teams must implement the right security defaults to ensure their security.

Cluster compromise can happen when there are misconfigurations in core Kubernetes components. The most commonly misconfigured components on the Kubernetes control plan and nodes include the below:

- Kubelet – Check that the configuration is set up to deny anonymous authentication. Also, when communicating with Kubelets, authorization checks should always be performed. Set the Authorization mode to anything other than 'AlwaysAllow' in order to block unauthorized requests.
- Kube-apiserver – Inspect the internet accessibility of the API server in use and keep the Kubernetes API off of any public networks.

Performing CIS Benchmark scans and audits can help security teams focus on eradicating component misconfigurations. Using hosted services such as EKS, GKE, or AKS can help implement secure defaults and limit some of the options for component configuration.

10. Vulnerable Components

As Kubernetes clusters run vast amounts of third-party software, security teams will need to build a multi-tiered strategy to combat vulnerable components. Some best practices on how to do so are as follows:

- Track CVE databases – A key element in managing known and new vulnerabilities in Kubernetes is to stay up-to-date on CVE databases, security disclosures, and community updates. Security teams can use this intel to build actionable plans to implement regular patch management processes.
- Implement continuous scanning – Using a tool such as OPA Gatekeeper can be helpful in writing custom rules that work to uncover any vulnerable components within a Kubernetes cluster. Security teams can then track and document these findings to improve their security processes and policies.
- Minimize third-party dependencies – Third-party software must be thoroughly audited for overly permissive RBAC, low-level kernel access, and vulnerability disclosure records before deployment occurs.

Conclusion

Though Kubernetes is powerful, its adoption comes with the inevitable introduction of new risks into an environment's existing infrastructure and applications. Having a comprehensive approach to securing Kubernetes ensures security teams can address all types of vulnerabilities and risks that can affect the individual layers of a Kubernetes cluster.

Following the recommendations provided in this post can set businesses on the right path to hardening their Kubernetes environments and reduce its attack surface. Through best practices implementation, security teams managing Kubernetes can gain visibility into their environments and better control each layer of their Kubernetes deployment.

Read more about Cyber Security

- Detecting a Rogue Domain Controller – DCShadow Attack
- Surviving the Storm | Defending Against Cloud Misconfigurations, Vulnerabilities, and Insider Threats
- Threat Landscape | The Most Dangerous Cloud Attack Methods In The Wild Today
- Accelerating Your Cloud Security with Workload Protection
- EDR for Cloud Workloads Running on AWS Graviton
- Defending Cloud-Based Workloads: A Guide to Kubernetes Security