StackRox

Protecting Kubernetes:

The Kubernetes Attack Matrix and How to Mitigate Its Threats

StackRox

Protecting Kubernetes: The Kubernetes Attack Matrix and How to Mitigate Its Threats

Introduction

In the past five years, modern application architecture and compute infrastructure have been dramatically transformed by Kubernetes, the production-grade, open-source container orchestration system that has become the de facto standard for automating the deployment and management of cloud-native applications. At the same time, the adoption of Kubernetes and its use for business-critical operations drives new security challenges that must be appropriately understood and addressed, namely an expansion of the infrastructure attack surface that exposes organizations to new threats. Several of these threat vectors are detailed in the first-ever Kubernetes attack matrix published by Microsoft Azure and based on the MITRE ATT&CK framework of known tactics and techniques.

The publication of this attack matrix for Kubernetes marks an important milestone for the Kubernetes community and broader cloud-native ecosystem; it, along with related guidance from organizations such as the Cloud Native Computing Foundation (CNCF), National Institute of Standards and Technology (NIST), and Center for Internet Security (CIS), reflect the growing need of Kubernetes end users across organizations of every size and industry to ensure adequate security for their Kubernetes environments in light of new, emerging threats. The Kubernetes community also recognizes that unique threats to Kubernetes exist and has undertaken several efforts, including commissioning an inaugural threat model and security audit last year.

The attack matrix also reflects recognition by the ecosystem that a “container-centric” approach to security - meaning only focusing on securing containers - is not enough to protect organizations from the variety of threats that target Kubernetes. Organizations must adopt strategies that ensure comprehensive security at the orchestrator level, which requires solutions that are purpose-built for Kubernetes environments and preferably “Kubernetes-native.”

The Kubernetes Attack Surface

The attack surface within a Kubernetes cluster consists of two major components that must be protected: (1) the Kubernetes control plane which is used to manage operations throughout the cluster, and (2) the cluster's worker nodes which run an organization's applications in pods. Each of these further has more specific components that can be exposed to various types of threat vectors.

Major Kubernetes Components

The Kubernetes control plane's components include the Kubernetes API server (kube-apiserver), kube-scheduler, kube-controller-manager, cloud-controller-manager, and etcd, each of which can be run on any node in the cluster. Together, the first three - kube-apiserver, kube-scheduler, and kube-controller-manager - are also collectively referred to as the Kubernetes Master. Compromise of control plane components may easily result in complete compromise of a cluster, considering that the control plane is designed to make global decisions regarding a cluster's operations.

The Kubernetes components that worker nodes run are the kubelet, kube-proxy, and the container runtime, all of which run on every worker node within the cluster. Compromise of worker node components can result in direct compromise of the containerized application components running within the pods on the particular node.

The inaugural Kubernetes security audit was based on a threat model that spanned the eight Kubernetes components referenced.

How Kubernetes threat vectors differ from traditional ones

The number and complexity of various components across a cluster means that organizations running Kubernetes are exposed to a wide variety of threats specific to the platform, each with a different scope. These threats continue to evolve as the Kubernetes community continues to innovate and expand the platform's capabilities. Some examples of Kubernetes-specific threats that have been observed in running environments to date include:

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

Kubernetes threat vectors differ markedly from traditional ones in the sense that attacks can be highly distributed, dynamic, and fast-moving, reflecting the nature of containers and microservices themselves. Application components running in pods are no longer tied to specific machines as traditional applications are. They may also be ephemeral, unlike traditional monolithic applications. These attributes can be exploited by adversaries to execute attacks with greater speed and scale, and within shorter time spans, than ever before.

What StackRox's research reveals about Kubernetes threats

From research conducted by its security teams, StackRox has previously found that Internet-exposed applications running on Kubernetes face many of the same threats that affect non-containerized applications. This research focused on threats directed to applications, but adopting secure configurations and practices for Kubernetes infrastructure is critical to minimizing the effects of a compromise in any given Kubernetes application.

To conduct this research, StackRox operated large Google Kubernetes Engine (GKE) clusters around the world and exposed hundreds of containerized applications to traffic from the Internet for nearly five months. These applications primarily utilized common images available from popular image registries such as Docker Hub and included web servers, databases, and developer tools. The StackRox team also deployed applications with known vulnerabilities and deliberately weak configurations such as disabled authentication.

StackRox's research observations revealed common adversary techniques that included the following:

- Injection attempts to download a file into /tmp/ with a short alphabetical filename.
- Attempted downloads using wget.
- Intrusion attempts frequently occurring on well-known web ports.
- Attempted commands that would occasionally, if successfully executed, provide an adversary with additional targeting data, such as the operating system. Further commands would attempt to download a binary, usually to exfiltrate data or launch other forms of command-and-control traffic.

Many of these observations align with the tactics and techniques that are described in the Kubernetes attack matrix.

Overview of the Kubernetes Attack Matrix

The Kubernetes attack matrix published by Microsoft Azure is based on the well-known MITRE ATT&CK framework, a comprehensive matrix of adversary tactics and techniques that is intended to help organizations classify attacks and assess their level of risk. The ATT&CK framework has evolved since its initial release in 2015, and today the ATT&CK Matrix for Enterprise consists of a dozen different tactics that each encompass a substantial number of attacker techniques. "Tactics" are considered generalized categories for varied types of attacker objectives, while "techniques" reference specific actions that attackers may take to accomplish the given tactic. The documented techniques within the ATT&CK framework are based on observations from attacks "in the wild."

StackRox

The Kubernetes attack matrix extends the ATT&CK framework for the first time to Kubernetes to describe a total of 40 different techniques that fall under nine different tactics.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Impact
Using cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8s secrets	Access the K8s API server	Access cloud resources	Data destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Resource hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod/container name similarity	Access container service account	Network mapping	Cluster internal networking	Denial of service
Application vulnerability	Application exploit (RCE)		Access cloud resources	Connect from proxy server	Application credentials in configuration files	Access K8s dashboard	Application credentials in configuration files	
Exposed dashboard	SSH server running inside container					Instance Metadata API	Writable volume mounts on the host	
							Access K8s dashboard	
							Access Tiller endpoint	

Note: Clicking on any of these terms will take you to the discussion of that attack technique in the paper.

Kubernetes Attack Matrix Tactics and Techniques

The following sections detail each of these 40 techniques and include guidance on security best practices to mitigate these threats. It also highlights whether the security controls that should be primarily used and configured to adequately protect against these threats reside within (1) Kubernetes itself, (2) cloud provider platform services, or (3) other cloud-native tooling such as container image registries, package management systems, or others. To comprehensively secure Kubernetes, organizations must protect both the infrastructure systems that underlie it and those that directly interface with it.

This document also describes, where applicable, how the StackRox Kubernetes Security Platform protects organizations from the threats that make up the Kubernetes attack matrix. StackRox has the industry's only Kubernetes-native security architecture, which delivers unique capabilities in identifying and stopping the representative exploits described within the Kubernetes attack matrix.

Summary

Of the 40 threats enumerated in the Kubernetes attack matrix, 20 threats can be mitigated by using or configuring security controls in Kubernetes itself. This high percentage reflects how native security controls in Kubernetes provide an extensive starting point for protecting Kubernetes environments. Three can be addressed by using or configuring controls in cloud provider services. Six techniques involve setting up protections in other cloud-native tooling, given that toolchains make up part of the overall attack surface as well. Three threat vectors warrant configuring controls in both Kubernetes and other tooling. One other is best mitigated by configuring controls across Kubernetes, cloud providers, and other tooling. Of the 40 threats, 11 can be addressed by configuring security controls in both Kubernetes and cloud provider services. This is unsurprising given that several of the techniques are specific to Kubernetes environments running on cloud provider platforms.

For several of these threats, mitigation requires configuring controls in a combination of areas (e.g., Kubernetes and cloud provider services) to accomplish security best practices. StackRox provides built-in policies, automated capabilities, or features that directly mitigate 33 of the threats, spanning all tactics in the attack matrix.

Primary area to configure security controls

1

Tactic #1: Initial Access

Tactic #1: Initial Access

The first tactic, or category of attacker techniques, in the Kubernetes attack matrix is Initial Access. This tactic references the attacker's objective of gaining access to Kubernetes clusters by either compromising a component within the Kubernetes control plane that handles cluster management or various resources within the cluster – for example, a running application component on a worker node.

StackRox helps protect Kubernetes environments from these threats by enforcing policies on images that can be used, including validating that they originated from trusted image registries; discovering vulnerabilities in container images, running Kubernetes Deployments, and Kubernetes system components; automatically configuring Kubernetes Network Policies to restrict traffic between pods; and monitoring RBAC configurations to identify user and service accounts with elevated privileges on the cluster.

Technique 1.1: Using Cloud Credentials

Issue

An attacker who gains access to cloud provider account credentials can use those to compromise entire Kubernetes clusters, whether running on managed Kubernetes platforms offered by cloud providers or administered by the customer on cloud compute instances.

Best Practice for Mitigation

Primary area to configure security controls: Cloud Provider

Organizations can mitigate this threat by configuring cloud provider identity and access management (IAM) services to restrict access to these account credentials, applying the principle of least privilege, and also ensuring that these credentials are rotated regularly. The use of shared credentials should be minimized, meaning IAM privileges should be granted to individual users rather than relying on shared accounts.

Technique 1.2: Compromised images in registry

Issue

Unsafe or compromised images that were either (1) downloaded from a public image registry or (2) which were added to a private registry and used to launch containers can result in an attacker gaining access to a cluster.

Real-world example: *Backdoored container images that contained malicious code for mining the Monero cryptocurrency were publicly available on Docker Hub and downloaded more than five million times.*

Best Practice for Mitigation

Primary area to configure security controls: Other - Image registry

To protect against this threat, organizations can implement multiple controls that include only allowing images from trusted registries to be used, limiting the use of images from public registries such as Docker Hub, restrict access to trusted and/or private registries, and requiring developers to only use trusted base images when building containerized applications.

How StackRox Helps

The StackRox Kubernetes Security Platform can automatically enforce policies on what images can be used to launch containers and can also identify security issues, including vulnerabilities and problematic packages, in image layers separate from the underlying base operating system (OS image. It provides visibility and analysis by enumerating these contents of images. StackRox also automatically monitors runtime activity to detect known bad processes that execute within pods such as processes meant to carry out cryptomining.

Technique 1.3: Kubeconfig file

Issue

kubeconfig is used by kubectl and contains the location and credentials of clusters. A compromised client could issue commands to gain unauthorized access to this file.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Organizations should ensure the Kubernetes clients that end users are running to issue these commands, including kubectl, are updated to the latest version available. They should also configure the Kubernetes API server to use a third-party authentication provider rather than relying on static client certificates or other credentials that can be easily exploited.

Technique 1.4: Application vulnerability

Issue

Containerized applications that are public-facing can allow initial attacker access, and if these containers have vulnerabilities, they leave organizations susceptible to exploits. For example, a remote code execution (RCE) vulnerability could be exploited and since service account credentials are mounted to containers by default in Kubernetes, these credentials could then be used to make requests to the Kubernetes API server. More generally, exploiting a vulnerability may also allow an attacker to reach other applications, access the kubelet read-only API server, access sensitive data stored on cloud provider instance metadata servers, cause a denial of service attack, or other malicious activities.

Real-world example: *CVE-2014-3120 is a RCE vulnerability in ElasticSearch that has been observed to have been used to breach a Docker environment in the wild.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

Organizations should start by scanning their container images for vulnerabilities and configuring Kubernetes Network Policies to restrict external access to pods within a cluster. Admission control can be used to prevent containers with known high-severity vulnerabilities from launching.

Additionally service account permissions should be restricted using Kubernetes Role-Based Access Control (RBAC) according to the principle of least privilege. Service accounts should be unique to workloads, and administrators should ensure that workloads do not mount service account tokens unless they are needed to access the Kubernetes API server.

Pods should be further secured by only allowing non-root users, enabling a read-only root file system, and minimizing access to the underlying host.

Cloud Provider

If running on a cloud provider platform, controls such as Google Kubernetes Engine's (GKE) metadata concealment, Workload Identity, or equivalent options should be enabled to prevent compromised pods from being leveraged to access cloud resources.

How StackRox Helps

StackRox scans container images for vulnerabilities as they are built and provides the option to fail CI/CD builds based on whether images violate specific policies including if certain vulnerabilities are present. The platform also enables users to quickly discover new vulnerabilities in running Kubernetes Deployments. StackRox also enables dynamic admission control to prevent containers with specific vulnerabilities from launching.

StackRox can automatically configure Kubernetes Network Policies based on observed network traffic to restrict external access to pods. It also automatically monitors Kubernetes RBAC configurations to help users identify risks and security weaknesses related to access settings.

Finally, StackRox can enforce policies to secure pods such as only allowing non-root users and configuring read-only root file systems.

Technique 1.5: Exposed dashboard

Issue

The Kubernetes Dashboard is a web-based user interface that can be used for managing cluster resources. It exposes an internal endpoint and if accessible publicly, it can allow unauthenticated cluster management. Certain versions of Kubernetes platforms may deploy the Dashboard by default.

Real-world example: *The car company Tesla experienced a breach of its Amazon Web Services (AWS) infrastructure due to a Kubernetes Dashboard that was exposed to the Internet and did not require authentication. The Dashboard had elevated privileges on the cluster, allowing attackers to mine cryptocurrency and to read secret values, including AWS credentials.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes

If the Dashboard is not needed, administrators should ensure that it is deleted from the environment entirely or disabled (which is now generally the case for many Kubernetes platforms). If the Dashboard is needed and is deployed, then do not grant it elevated service account privileges, remove any bindings to its service account, and block ingress traffic using Kubernetes Network Policies.

How StackRox Helps

StackRox provides a built-in policy to alert when the Kubernetes Dashboard is deployed. It also automatically monitors RBAC privileges on service accounts and can identify whether elevated privileges have been granted to the Dashboard. It can also ensure incoming traffic to the Dashboard is blocked by configuring Kubernetes Network Policies.

2

Tactic #2: Execution

Tactic #2: Execution

The second tactic in the Kubernetes attack matrix is Execution, which focuses on an attacker running code within a Kubernetes cluster to achieve his or her objectives. Malicious code could be executed by gaining access to a running pod, starting a new pod, or exploiting an application vulnerability, or other means.

StackRox helps secure organizations from these threats with capabilities that include the following: detecting and alerting on execution of anomalous, suspicious, or malicious processes; alerting on the use of suspicious tools; incorporating tools that may be useful to attackers into its automated risk profiling; enforcing secure pod configurations; and monitoring RBAC privileges.

Technique 2.1: Exec into container

Issue

Kubectl is a command line tool for managing Kubernetes clusters. 'kubectl exec' allows a user to execute a command in a container. Attackers with permissions could run 'kubectl exec' to execute malicious code and compromise resources within a cluster.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Administrators can mitigate this threat by restricting RBAC access to the "pods/exec"' resource according to the principle of least privilege. Kubernetes RBAC can be used to reference both resources and subresources. Also, pod configurations can be hardened to limit what an attacker can accomplish by removing unnecessary privileges and ensuring pods run with a read-only root file system.

How StackRox Helps

StackRox helps mitigate threats related to 'kubectl exec' by monitoring Kubernetes RBAC configurations for users and service accounts with excess privileges. It automatically monitors runtime activity in every container to detect and alert on anomalous, suspicious, or malicious processes that may execute within a container as a result of an attacker issuing a command. StackRox also alerts on the use of suspicious tools and enforces policies for secure pod configurations.

Technique 2.2: bash/cmd inside container

Issue

This technique references the potential for attackers with permissions to run a bash script inside a container to execute malicious code and compromise cluster resources.

Best Practice for Mitigation

Primary area to configure security controls: Cloud Provider

Organizations can best reduce their exposure to this threat by minimizing access to container hosts.

How StackRox Helps

StackRox helps mitigate this threat in multiple ways. StackRox's risk profiling automatically identifies containers with tools that are potentially useful to attacker, including bash. It also alerts on the use of suspicious tools as well as monitors, detects, and alerts on concerning runtime activity such as execution of abnormal or unexpected processes within containers.

Technique 2.3: New container

Issue

Existing, running pods are not the only resources that can be utilized by threat actors, and this technique highlights the ability of attackers with permissions to launch new pods as well as Deployments, DaemonSets, ReplicaSets, and other resources that reference controllers to execute their malicious code and compromise cluster resources.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Organizations can implement protections for this technique by controlling and restricting RBAC permissions to create pods and/or abstractions (such as Deployments, DaemonSets, ReplicaSets, and others) that also create pods.

How StackRox Helps

StackRox helps organizations limit RBAC permissions according to the principle of least privilege by automatically monitoring RBAC settings for users and service accounts and identifying ones with overly excessive privileges on clusters. StackRox also analyzes image contents, pod configurations, and runtime activity within pods and gives organizations the ability to optionally block non-compliant container deployments or delete suspicious pods.

Technique 2.4: Application exploit (RCE)

Issue

Containers with a vulnerability that allows for remote code execution can be exploited by an attacker to run malicious code and since service account credentials are mounted to containers by default, these credentials can be used to make requests to the Kubernetes API server that can lead to cluster compromise. Exploiting a vulnerability may also allow an attacker to compromise other resources, access the kubelet read-only API server, access sensitive data stored on cloud provider instance metadata servers, cause a denial of service attack, or undertake other malicious activities.

Real-world example: *Docker containers running ElasticSearch with the RCE vulnerability CVE-2014-3120 were observed to have been breached.*

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Administrators should ensure that applications use unique service accounts and that service account tokens are not mounted if an application does not require access to the Kubernetes API server. Container images should be scanned for vulnerabilities, including RCE vulnerabilities, and users should ensure that running Kubernetes Deployments do not have vulnerabilities that would allow for code execution.

How StackRox Helps

StackRox scans container images for vulnerabilities and enforces policies on how images with specific vulnerabilities can be used and provides a Kubernetes admission controller to prevent containers with vulnerabilities from launching. It makes it easy to discover vulnerabilities in existing Deployments and also automatically monitors RBAC configurations to help determine whether access to the Kubernetes API server is scoped down to as few privileges as required.

Technique 2.5: SSH server running inside container

Issue

This technique under Execution arises when an SSH server is running inside a container, which could allow an attacker who obtains credentials to that container through other means to gain remote access to the container to run malicious code and compromise resources.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Other - tooling

Kubernetes

In Kubernetes, administrators should limit service exposure and apply Kubernetes Network Policies to restrict network traffic and prevent unintended access to a container that is running an SSH server. Pod configurations should also be hardened to prevent SSH servers from being added at runtime.

Other - Tooling

Additionally, organizations should audit any installations of SSH server software that exist in container images.

How StackRox Helps

StackRox delivers built-in policies within its platform that identify Deployments with SSH ports exposed and pods that execute 'sshd' processes. The StackRox platform's user interface also provides a visualization of network traffic flows to and from containers, making it easy to identify unexpected communication with containers that are running SSH servers.

3

Tactic #3: Persistence

Tactic #3: Persistence

The third tactic in the Kubernetes attack matrix is Persistence. This tactic groups together techniques that are aimed at enabling an attacker to maintain a presence within a Kubernetes cluster beyond initial access through actions such as taking advantage of Kubernetes controllers, mounting a file to a container, or running recurring Kubernetes Jobs.

StackRox helps guard against these attack vectors by incorporating customizable policy-driven admission control into its platform to enforce security policies on container deployments, enforcing policies on pod configurations, analyzing container image contents, monitoring RBAC configurations for users and service accounts, and collecting runtime activity of all pods.

Technique 3.1: Backdoor container

Issue

Similar to Technique 2.3 (New Container) under the Execution tactic, this technique highlights an attacker's ability to potentially utilize Kubernetes controllers to ensure a container is always running somewhere in the cluster with the ability to execute malicious code.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Organizations can implement protections for this technique by controlling and restricting RBAC permissions to create pods and/or abstractions (such as Deployments, DaemonSets, ReplicaSets, and others) that also create pods.

How StackRox Helps

StackRox helps organizations limit Kubernetes RBAC permissions according the principle of least privilege by monitoring RBAC settings for users and service accounts and identifying ones with overly excessive privileges on clusters. StackRox also analyzes image contents, pod configurations, and runtime activity within pods and gives organizations the ability to optionally block non-compliant container deployments or delete suspicious pods.

Technique 3.2: Writable hostPath mount

Issue

With this technique, the hostPath volume mounts a file or directory to the container, which would allow an attacker to persist on the container host.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

In Kubernetes, users can apply Pod Security Policies to limit the file paths that can be mounted using a host mount or disallow host mounts completely (note that Persistent Volume Claims bypass this policy). They can also mark any required host paths as read-only whenever possible.

Cloud Provider

When configuring cloud provider environments, teams can limit node lifetimes by ensuring reverse uptime of 24 hours or less and automatically provision new nodes to replace them.

How StackRox Helps

The StackRox platform helps mitigate this threat by delivering dynamic policy-driven admission control as part of its platform that enables organizations to automatically enforce security policies, including limitations on host mounts and their writability, before containers are ever deployed into Kubernetes clusters.

Technique 3.3: Kubernetes CronJob

Issue

A Kubernetes Job creates one or more pods to accomplish a specific task, and a CronJob creates Jobs on a recurring schedule. An attacker can take advantage of this Kubernetes object to schedule Jobs to run containers that execute malicious code within a cluster.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Organizations can take steps to control RBAC permissions to create Jobs and pods and/or the abstractions (such as Deployments, DaemonSets, ReplicaSets, and others) that also create pods.

How StackRox Helps

StackRox helps mitigate against the threat of an attacker leveraging Kubernetes CronJobs for malicious activities by monitor pods running within a cluster, including all runtime activity, as well as monitoring native Kubernetes objects including CronJobs, even if containers are not currently running.

4

Tactic #4: Privilege Escalation

Tactic #4: Privilege Escalation

The fourth tactic in the Kubernetes attack matrix is Privilege Escalation, which encompasses techniques that enable an attacker to gain additional privileges that can be used to take more actions within the cluster and/or grant access to a wider scope of resources. These techniques include accessing or running a privileged container, taking advantage of roles with broad administrative privileges, and gaining access to cloud resources.

Some ways that StackRox provides protections against these privilege escalation techniques are: enforcing policies on whether privileged containers are allowed to run; configuring Kubernetes Network Policies to segment and restrict traffic between, to, and from pods; and monitoring RBAC settings for excess privileges.

Technique 4.1: Privileged container

Issue

In this first technique under Privilege Escalation, an attacker who gains access to a privileged container or has the ability to start a new container that is privileged will have all the capabilities of the host and can therefore gain access to host resources or compromise other containers running on the same host.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Organizations should implement several security practices to address the possibility of privilege escalation. First, they should limit the number of privileged containers that run within clusters, only allowing them when necessary. They should further configure Kubernetes Network Policies to segment network access to privileged containers and restrict RBAC privileges to create new Deployments or pods (in order to prevent a compromised pod's service account credentials from being used to start a new privileged container).

How StackRox Helps

StackRox helps address the risks associated with privileged containers to enforce policies on whether privileged containers are allowed to run. It helps users visualize, simulate, and automatically generate and configure Kubernetes Network Policies. StackRox also monitors RBAC privileges to ensure that users or service accounts are not unnecessarily able to start privileged containers.

Technique 4.2: Cluster-admin role binding

Issue

The risk created by this threat vector is that users who are granted the cluster-admin role then have full access to the cluster and, consequently, have the ability to accomplish complete cluster compromise. Additionally, users who have RBAC permissions to create role bindings can create a binding to the cluster-admin role or other roles with high privileges.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Administrators should configure Kubernetes RBAC to limit who in their organization has the cluster-admin role and who can create role bindings. They should also replace uses of the cluster-admin role with users, groups, or service accounts that have more targeted, scoped-down permissions.

How StackRox Helps

StackRox helps mitigate this threat by automatically monitoring RBAC configurations and identifying users, groups, or service accounts with elevated privileges, including the cluster-admin role specifically.

Technique 4.3: hostPath mount

Issue

Similar to Technique 3.2 (Writable hostPath mount), the hostPath volume mounts a file or directory to the container, which would allow an attacker to gain access to resources or compromise other containers running on the same host.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

In Kubernetes, users can apply Pod Security Policies to limit the file paths that can be mounted using a host mount or disallow host mounts completely (note that Persistent Volume Claims bypass this policy). They can also mark any required host paths as read-only whenever possible.

Cloud Provider

When configuring cloud provider environments, teams can limit node lifetimes by ensuring reverse uptime of 24 hours or less and automatically provision new nodes to replace them.

How StackRox Helps

The StackRox platform helps mitigate this threat by delivering dynamic policy-driven admission control as part of its platform enables organizations to enforce security policies, including limitations on host mounts and their writability, before containers are ever deployed into Kubernetes clusters.

Technique 4.4: Access cloud resources

Issue

Organizations running Kubernetes clusters in cloud environments, including using cloud provider-managed Kubernetes services, must be aware of additional techniques that adversaries may seek to leverage. In this case, this technique involves an adversary gaining access to a single container, which could allow access to other cloud resources.

Real-world example: *In Azure Kubernetes Service (AKS), each node contains a "service principal" credential that is stored in /etc/kubernetes/azure.json. AKS uses this service principal to create and manage Azure resources that are needed for the cluster operation. By default, the service principal has contributor permissions in the cluster's Resource Group. Attackers who get access to this service principal file (by hostPath mount, for example) can use its credentials to access or modify the cloud resources. Note that this concept of a service principal is specific to AKS and is not directly applicable elsewhere.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Many of the security practices that organizations can implement to mitigate this threat overlap with other techniques that target access to specific types of cloud resources, e.g., instance metadata.

Kubernetes

In Kubernetes, users can apply Pod Security Policies to limit the file paths that can be mounted using a host mount or disallow host mounts completely (note that Persistent Volume Claims bypass this policy). They can also mark any required host paths as read-only whenever possible.

Cloud Provider

If running on a cloud provider platform, controls such as Google Kubernetes Engine's (GKE) metadata concealment, Workload Identity, or equivalent options should be enabled to prevent compromised pods from being leveraged to access cloud resources. Teams can also limit node lifetimes by ensuring reverse uptime of 24 hours or less and automatically provision new nodes to replace them.

How StackRox Helps

The StackRox platform helps mitigate attacker access to cloud resources with several capabilities. StackRox scans container images for vulnerabilities as they are built and enables users to quickly discover new vulnerabilities in running Kubernetes Deployments.

StackRox also enables dynamic admission control to prevent containers with specific vulnerabilities from launching and to enforce broader security policies, including limitations on host mounts and their writability, before containers are ever deployed into Kubernetes clusters. It enforces policies to secure pods such as only allowing non-root users and configuring read-only root file systems.

StackRox visualizes, simulates, and automatically configures Kubernetes Network Policies based on observed network traffic to restrict external access to pods. It also monitors Kubernetes RBAC configurations to help users identify risks and security weaknesses related to access settings.

5

Tactic #5: Defense Evasion

Tactic #5: Defense Evasion

The fifth tactic in the Kubernetes attack matrix is Defense Evasion, which groups techniques focused on concealing adversary actions intended to avoid detection such as deleting evidence of an attacker's presence or obfuscating how access to a resource was gained.

StackRox helps mitigate these Defense Evasion threats by monitoring all pods, including those not created by using a Kubernetes controller, and identifying standalone pods directly in the StackRox user interface.

Technique 5.1: Clear container logs

Issue

With this technique, an attacker may delete logs from the container runtime or operating system that capture the attacker's activity within containers.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes, Cloud Provider, and Other - container runtime and operating system

Kubernetes

Within Kubernetes, organizations can mitigate this threat by minimizing container access to nodes by restricting host mounts (also see Techniques 3.2 "Writable hostPath mount" and 4.3 "hostPath mount" for reference).

Cloud Provider

Within the cloud provider environment, organizations should minimize access to cluster nodes to mitigate this threat.

Other - Container runtime and operating system

Organizations should continuously send collected logs from the container runtime and operating system to a secure, external logging service, security information and event management (SIEM) system, or persistent storage.

Technique 5.2: Delete Kubernetes events

Issue

Another technique an attacker may use to avoid detection is to delete Kubernetes audit logs, which provide a chronological record of security-relevant activities taken by users or system components within a Kubernetes cluster.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

Organizations should configure Kubernetes audit logging to continuously send events to an external logging service outside clusters. Additionally, administrators should minimize container access to underlying nodes, especially in the Kubernetes control plane, by restricting host mounts (also see Techniques 3.2 “Writable hostPath mount” and 4.3 “hostPath mount” for reference).

Cloud Provider

If running clusters using a cloud provider-managed Kubernetes service, organizations should use activity logging feature available. Administrators should also further minimize user access to underlying nodes.

Technique 5.3: Pod / container name similarity

Issue

This threat arises from the possibility that pods created by controllers such as Deployments or DaemonSets may have a random suffix in their names. An attacker could use a random suffix to obfuscate the presence of an unauthorized pod within the cluster that could be used to execute malicious code or gain access to additional resources.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes

Administrators should mitigate the risk of this threat by following the principle of least privilege with regard to access control and limiting RBAC permissions on pod creation APIs.

How StackRox Helps

StackRox helps protect against this attack technique by monitoring all pods, including those not created by using a Kubernetes controller. StackRox also identifies standalone pods (those not created by a controller) in its platform.

Technique 5.4: Connect from Proxy server

Issue

To conceal their network origin, attackers may employ this technique by using proxy servers or anonymous networks to hide their IP address.

Best Practice for Mitigation

Primary areas to configure security controls: Cloud Provider

Organizations can mitigate this threat by configuring controls within the cloud provider such as applying network access restrictions to the Kubernetes API server using managed Kubernetes service controls, if applicable, or cloud network firewall rules. Note that these practices can also mitigate Kubernetes API server vulnerabilities or misconfigurations such as CVE-2018-1002105.

6

Tactic #5: Credential Access

Tactic #6: Credential Access

The sixth tactic in the attack matrix is Credential Access. The techniques under this tactic focus on stealing sensitive credentials such as application secrets, passwords, and tokens that may be used by either users or service accounts. These credentials can subsequently be used to gain access to resources that include applications, cluster resources (e.g., pods, controllers, or other Kubernetes objects), cloud resources, and others.

StackRox mitigates threats related to Credential Access in several ways. It monitors Kubernetes RBAC settings for overly permissive access to secrets, enforces security policies on Deployments' access to the Kubernetes API server, and also automatically discovers potential secrets that are passed in environment variables.

Technique 6.1: List Kubernetes secrets

Issue

Kubernetes Secrets enable organizations to store sensitive credentials such as password, tokens, and keys. An attacker may utilize this technique to gain permission to secrets from the Kubernetes API service that may include credentials for key services such as a database service.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Organizations can achieve protections for this threat by limiting access to secrets, including permissions on users and service accounts that are able to list secrets, and rotating secrets regularly. They should also use Kubernetes Namespaces to avoid accidental secret disclosures since secrets cannot be mounted across namespaces and because Kubernetes RBAC access is delineated along namespace boundaries.

How StackRox Helps

StackRox helps provide protections for this threat by automatically monitoring RBAC settings and identifying users and accounts with overly permissive access to secrets.

Technique 6.2: Mount service principal

Issue

With this technique, an attacker can leverage access to a single container to gain access to other cloud resources. Note that there is substantial overlap with Techniques 3.2 "Writable hostPath mount", 4.3 "hostPath mount", and 4.4 "Access cloud resources" for reference.

Real-world example: *In Azure Kubernetes Service (AKS), each node contains a "service principal" credential that is stored in /etc/kubernetes/azure.json. AKS uses this service principal to create and manage Azure resources that are needed for the cluster operation. By default, the service principal has contributor permissions in the cluster's Resource Group. Attackers who get access to this service principal file (by hostPath mount, for example) can use its credentials to access or modify the cloud resources. Note that this concept of a service principal is specific to AKS and is not directly applicable elsewhere.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

In Kubernetes, users can apply Pod Security Policies to limit the file paths that can be mounted using a host mount or disallow host mounts completely (note that Persistent Volume Claims bypass this policy). They can also mark any required host paths as read-only whenever possible.

Cloud Provider

When configuring cloud provider environments, teams can limit node lifetimes by ensuring reverse uptime of 24 hours or less and automatically provision new nodes to replace them.

How StackRox Helps

The StackRox platform helps mitigate this threat by delivering dynamic policy-driven admission control as part of its platform that enables organizations to enforce security policies, including limitations on host mounts and their writability, before containers are ever deployed into Kubernetes clusters.

Technique 6.3: Access container service account

Issue

By default, a service account is mounted to every pod in a Kubernetes cluster, which allows containers to send requests to the Kubernetes API server. An attacker who gains access to a pod can obtain the service account token. If Kubernetes RBAC is not enabled, the service account token will grant the attacker full access to the cluster. If RBAC is enabled, service account privileges are determined by role bindings. If these grant elevated privileges, an attacker can make requests to the Kubernetes API server to compromise cluster resources.

Best Practice for Mitigation

Primary area to configure security controls: Kubernetes

Organizations can mitigate this threat vector by configuring Kubernetes RBAC and adopting a least privilege model for service accounts and their role bindings. For example, the use of the cluster-admin role should be highly restricted (see Technique 4.2 “Cluster-admin role binding”).

How StackRox Helps

StackRox helps guard against this technique by monitoring Kubernetes RBAC configurations and enforcing security policies on pod access to the Kubernetes API server.

Technique 6.4: Application credentials in configuration files

Issue

Secrets are often stored as environment variables in pod configurations and Kubernetes configuration files. Access to these configurations would allow a malicious actor to steal these secrets and access cluster resources.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Other - Open source tooling

Kubernetes
Organizations should avoid the risks of this threat vector by implementing checks and processes to ensure secrets are not passed in Kubernetes manifests, which are used to create, modify and delete Kubernetes resources.

Other - Open source tooling
Organizations can consider using open source tools to identify secrets that have been added to source control.

How StackRox Helps

StackRox helps mitigate this technique by providing a built-in policy that identifies potential secrets being stored in environment variables.

7

Tactic #7: Discovery

Tactic #7: Discovery

The seventh tactic in the Kubernetes attack matrix is Discovery. The techniques in this category are intended to help an attacker effectively explore a Kubernetes environment to achieve lateral movement and gain access to a wider scope of resources with or beyond the cluster. They include ways to gain access to the Kubernetes API server or the Kubelet API, map the cluster network, or compromise resources via the Kubernetes Dashboard or cloud instance metadata.

StackRox helps address these threats by visualizing and configuring Kubernetes Network Policies, including blocking access to the kubelet port, and monitoring Kubernetes RBAC privileges.

Technique 7.1: Access the Kubernetes API server

Issue

This technique focuses on the Kubernetes API server, a critical component that serves as the front end of the Kubernetes control plane and exposes the Kubernetes API. An attacker who gains access to the Kubernetes API server can retrieve information about a cluster's resources.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

Administrators should ensure the Kubernetes API server is configured securely and limit (1) which users and service accounts have access to the Kubernetes, and (2) their permissions. Enable and configure Kubernetes RBAC to limit which users and service accounts have access to the Kubernetes API server and their permissions.

Other - Open source tooling

Administrators should restrict external cluster access to trusted source IP addresses only.

How StackRox Helps

StackRox mitigates risks associated with adversaries accessing the Kubernetes API server by analyzing RBAC permissions to limit the users and service accounts that have privileges to retrieve information about cluster resources.

Technique 7.2: Access Kubelet API

Issue

This technique exploits the Kubelet, an agent that is installed on every Kubernetes node and exposes a read-only API service that does not require authentication on TCP port 10255. An attacker with network access to the host can query the Kubelet API to discover running pods on the host as well as information about the host such as CPU and memory consumption.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes

Organizations can mitigate this threat by configuring Network Policies to block pod access to the Kubelet port or restrict other sensitive network egress.

Technique 7.3: Network Mapping

Issue

This technique takes advantage of the fact that, by default, Kubernetes does not restrict network traffic between pods. An attacker who gains access to a single pod can map the cluster network to discover other running pods/applications.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes

Organizations can mitigate this threat by enabling and configuring Kubernetes Network Policies to restrict and segment traffic between pods, preventing an attacker from discovering every pod running in a cluster.

How StackRox Helps

StackRox helps protect against network mapping by monitoring active network traffic between pods and automatically generating and configuring Network Policies to restrict communications to only what is necessary for application components to operate.

Technique 7.4: Access Kubernetes Dashboard

Issue

By default, Kubernetes does not restrict network traffic between pods. An attacker who gains access to a single pod can subsequently access the Kubernetes Dashboard and retrieve information about the cluster.

Real-world example: *The car company Tesla experienced a breach of its Amazon Web Services (AWS) infrastructure due to a Kubernetes Dashboard that was exposed to the Internet and did not require authentication. The Dashboard further had elevated privileges on the cluster and allowed attackers to obtain AWS credentials that were then utilized to repurpose the environment to mine cryptocurrency.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes

If the Dashboard is not needed, administrators should ensure that it is deleted from the environment entirely or disabled (which is now generally the case for many Kubernetes platforms). If the Dashboard is needed and deployed, then do not grant it elevated service account privileges, remove any bindings to its service account, and block ingress traffic using Kubernetes Network Policies.

How StackRox Helps

StackRox provides a built-in policy to alert when the Kubernetes Dashboard is deployed. It also monitors RBAC privileges on service accounts and can identify whether elevated privileges have been granted to the Dashboard. It can also ensure incoming traffic to the Dashboard is blocked by configuring Kubernetes Network Policies.

Technique 7.5: Instance Metadata API

Issue

Cloud providers expose metadata services to containers, which includes information about the environment, including the underlying hosts, or sensitive credentials. An attacker who is able to access this instance metadata can leverage it to access or compromise a broader set of either container or cloud resources.

Real-world example: *A server-side request forgery (SSRF) vulnerability was reported in Shopify infrastructure. This allowed an attacker to retrieve kube-env (which includes Kubelet credentials) from a cloud metadata service and could lead to full cluster compromise.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

In Kubernetes, organizations should ensure they configure egress Network Policies to restrict sensitive traffic including communication with cloud metadata services.

Cloud Provider

If running on a cloud provider platform, controls such as Google Kubernetes Engine's (GKE) metadata concealment, Workload Identity, or equivalent options should be enabled to prevent compromised pods from being leveraged to access cloud resources. Teams can also limit node lifetimes by ensuring reverse uptime of 24 hours or less and automatically provision new nodes to replace them.

How StackRox Helps

StackRox helps with this particular threat by visualizing Network Policies within its platform to help teams fully understand network details and traffic between pods throughout their Kubernetes environments.

8

Tactic #8: Lateral Movement

Tactic #8: Lateral Movement

Focus is on moving throughout the environment to gain access to other resources, including other containers, nodes, or cloud resources.

Technique 8.1: Access cloud resources

Issue

Organizations running Kubernetes clusters in cloud environments, including using cloud provider-managed Kubernetes services, must be aware of additional techniques that adversaries may seek to leverage. In this case, this technique involves an adversary gaining access to a single container, which could allow access to other cloud resources. Note this overlaps with Technique 4.4 "Access cloud resources" for reference.

Real-world example: *In Azure Kubernetes Service (AKS), each node contains a "service principal" credential that is stored in /etc/kubernetes/azure.json. AKS uses this service principal to create and manage Azure resources that are needed for the cluster operation. By default, the service principal has contributor permissions in the cluster's Resource Group. Attackers who get access to this service principal file (by hostPath mount, for example) can use its credentials to access or modify the cloud resources. Note that this concept of a service principal is specific to AKS and is not directly applicable elsewhere.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Many of the security practices that organizations can implement to mitigate this threat overlap with other techniques that target access to specific types of cloud resources, e.g., instance metadata.

Kubernetes

In Kubernetes, users can apply Pod Security Policies to limit the file paths that can be mounted using a host mount or disallow host mounts completely (note that Persistent Volume Claims bypass this policy). They can also mark any required host paths as read-only whenever possible.

Cloud Provider

If running on a cloud provider platform, controls such as Google Kubernetes Engine's (GKE) metadata concealment, Workload Identity, or equivalent options should be enabled to prevent compromised pods from being leveraged to access cloud resources. Teams can also limit node lifetimes by ensuring reverse uptime of 24 hours or less and automatically provision new nodes to replace them.

How StackRox Helps

The StackRox platform helps mitigate attacker access to cloud resources with several capabilities. StackRox scans container images for vulnerabilities as they are built and enables users to quickly discover new vulnerabilities in running Kubernetes Deployments.

StackRox also enables dynamic admission control to prevent containers with specific vulnerabilities from launching and to enforce broader security policies, including limitations on host mounts and their writability, before containers are ever deployed into Kubernetes clusters. It enforces policies to secure pods such as only allowing non-root users and configuring read-only root file systems.

Technique 8.2: Container service account

Issue

By default, a service account is mounted to every pod in a Kubernetes cluster, which allows containers to send requests to the Kubernetes API server. An attacker who gains access to a pod can obtain the service account token. If Kubernetes RBAC is not enabled, the service account token will grant the attacker full access to the cluster. If RBAC is enabled, service account privileges are determined by role bindings. If these grant elevated privileges, an attacker can make requests to the Kubernetes API server to compromise cluster resources. Also see Technique 6.3 “Access container service account” for reference.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes

Organizations can mitigate this threat vector by configuring Kubernetes RBAC and adopting a least privilege model for service accounts and their role bindings. For example, the use of the cluster-admin role should be highly restricted (see Technique 4.2 “Cluster-admin role binding”).

How StackRox Helps

StackRox helps guard against this technique by monitoring Kubernetes RBAC configurations and enforcing security policies on pod access to the Kubernetes API server.

Technique 8.3: Cluster internal networking

Issue

This threat vector arises since, by default, Kubernetes does not restrict network traffic between pods. An attacker who gains access to a single pod can communicate with and gain subsequent access to other running pods/applications.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes

Organizations can mitigate this threat by enabling and configuring Kubernetes Network Policies to restrict and segment traffic between pods, preventing an attacker from being able to reach every pod running in a cluster.

How StackRox Helps

StackRox helps protect against this technique by monitoring active network traffic and visualizing, simulating and automatically generating Kubernetes Network Policies to restrict communications to only what is necessary for application components to operate.

Technique 8.4: Application credentials in configuration files

Issue

Secrets are often stored as environment variables in pod configurations and Kubernetes configuration files. Access to these configurations would allow a malicious actor to steal these secrets and access cluster resources. Also see Technique 6.4 “Application credentials in configuration files” for reference.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Other - Open source tooling

Kubernetes

Organizations should avoid the risks of this threat vector by implementing checks and processes to ensure secrets are not passed in Kubernetes manifests, which are used to create, modify and delete Kubernetes resources.

Other - Open source tooling

Organizations can consider using open source tools to identify secrets that have been added to source control.

How StackRox Helps

StackRox helps mitigate this technique by providing a built-in policy that identifies potential secrets being stored in environment variables.

Technique 8.5: Writable mounts on the host

Issue

With this technique, the hostPath volume mounts a file or directory to the container, which would allow an attacker to persist on the container host. This overlaps with Technique 3.2 “Writable hostPath mount” for reference.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

In Kubernetes, users can apply Pod Security Policies to limit the file paths that can be mounted using a host mount or disallow host mounts completely (note that Persistent Volume Claims bypass this policy). They can also mark any required host paths as read-only whenever possible.

Cloud Provider

When configuring cloud provider environments, teams can limit node lifetimes by ensuring reverse uptime of 24 hours or less and automatically provision new nodes to replace them.

How StackRox Helps

The StackRox platform helps mitigate this threat by delivering dynamic policy-driven admission control as part of its platform enables organizations to enforce security policies, including limitations on host mounts and their writability, before containers are ever deployed into Kubernetes clusters.

Technique 8.6: Access Kubernetes Dashboard

Issue

The Kubernetes Dashboard is a web-based user interface that can be used for managing cluster resources. It exposes an internal endpoint and if accessible publicly, it can allow unauthenticated cluster management. Certain versions of Kubernetes platforms may deploy the Dashboard by default. See Technique 1.5 "Exposed Dashboard" and Technique 7.4 "Access Kubernetes Dashboard" for reference.

Real-world example: *The car company Tesla experienced a breach of its Amazon Web Services (AWS) infrastructure due to a Kubernetes Dashboard that was exposed to the Internet and did not require authentication. The Dashboard further had elevated privileges on the cluster and allowed attackers to obtain AWS credentials that were then utilized to repurpose the environment to mine cryptocurrency.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes

If the Dashboard is not needed, administrators should ensure that it is deleted from the environment entirely or disabled (which is now generally the case for many Kubernetes platforms). If the Dashboard needed and is deployed, then do not grant it elevated service account privileges, remove any bindings to its service account, and block ingress traffic using Kubernetes Network Policies.

How StackRox Helps

StackRox provides a built-in policy to alert when the Kubernetes Dashboard is deployed. It also monitors RBAC privileges on service accounts and can identify whether elevated privileges have been granted to the Dashboard. It can also ensure incoming traffic to the Dashboard is blocked by configuring Kubernetes Network Policies.

Technique 8.7: Access tiller endpoint

Issue

This technique is based on Tiller, which is the server-side component of earlier versions of Helm, a package manager for Kubernetes, and exposes an internal API endpoint that does not require authentication. It typically has high privileges and an attacker can use its service account to gain access to other containers.

Best Practice for Mitigation

Primary area to configure security controls: Other - Helm

Organizations can mitigate this threat by ensuring that they upgrade to the latest version of Helm for deploying their Kubernetes applications.

9

Tactic #9: Impact

Tactic #9: Impact

The ninth tactic in the attack matrix is focused on Impact. The techniques under this tactic are aimed at disrupting or destroying resources and activity within the target environment, or in other words, the ultimate goal of an attacker. These include techniques to achieve data destruction, resource hijacking or denial of service.

StackRox helps protect against this tactic and its techniques with a wide range of capabilities that include: scanning all container images that deploy into clusters; enforcing policies for image provenance; identifying Kubernetes vulnerabilities in clusters; monitoring Kubernetes configurations, RBAC access privileges, and runtime activity; and visualizing, simulating, and automatically generating Kubernetes Network Policies.

Technique 9.1: Data Destruction

Issue

This technique reflects the ability of an attacker to delete or remove resources in a Kubernetes cluster, including Deployments, configurations, nodes, pods, or other data.

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

Organizations can protect themselves from this threat vector by monitoring Kubernetes audit logs, which provide a chronological record of cluster events and restricting RBAC access according to the principle of least privilege.

Cloud Provider

Organizations should restrict access to Kubernetes cluster nodes and, if applicable, any cloud configuration APIs exposed by cloud provider-managed Kubernetes services.

How StackRox Helps

StackRox helps mitigate this threat by enforcing policies for and identifying weaknesses in Kubernetes configurations, monitoring RBAC access configurations, and analyzing runtime activity, such as process execution, within containers.

Technique 9.2: Resource Hijacking

Issue

With this technique, an attacker may hijack a Kubernetes resource for unintended purposes. One example of this is using compromised containers to run malicious tasks such as cryptomining.

Real-world example: *Microsoft Azure has disclosed the discovery of multiple campaigns, including a large-scale one, where attackers targeted Kubernetes clusters and were able to impact them by running cryptocurrency miners, otherwise known as "cryptojacking" attacks.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

Organizations can mitigate the threat of resource hijacking by configuring RBAC access to restrict the ability to create pods.

Cloud Provider

Within a cloud provider, organizations should take steps to restrict and minimize access to Kubernetes cluster nodes.

How StackRox Helps

StackRox helps protects organizations from resource hijacking by scanning all images that are used to deploy containers into the cluster and enforcing policies for image provenance. For example, many reported cryptojacking attacks have arisen from backdoored images stored in public registries containing malicious software. StackRox also monitors runtime activity with a built-in policy to detect cryptomining processes that execute.

Technique 9.3: Denial of service

Issue

With this technique, an attacker may seek to make services unavailable by impacting the availability of components within the Kubernetes control plane including the API server, cluster nodes, or application components in pods, or other resources.

Real-world example: *CVE-2019-1002100 is a vulnerability in the Kubernetes API server that allows users with certain write permissions on the Kubernetes API to make write requests that cause the API server to utilize excessive resources, resulting in a denial of service.*

Best Practice for Mitigation

Primary areas to configure security controls: Kubernetes and Cloud Provider

Kubernetes

Organizations can mitigate this threat by first ensuring they upgrade to the latest version of Kubernetes. Additionally, they can configure Kubernetes Network Policies to permit only necessary connections within the pod network.

Cloud Provider

Organizations should utilize cloud firewalls and other controls to restrict access to the cluster API endpoint and other sensitive, internally-consumed services to trusted IP addresses only.

How StackRox Helps

StackRox delivers protections to help mitigate this threat by automatically identifying Kubernetes vulnerabilities as well as visualizing, simulating, and automatically generating Kubernetes Network Policies to segment and restrict communication throughout the environment.

Conclusion

Kubernetes is enabling organization of every size across every industry in their efforts to modernize applications and infrastructure as part of strategic digital transformation efforts. Using Kubernetes, companies are now able to build and run applications at faster and at greater scale than previously possible. At the same time, Kubernetes creates a new threat environment that is complex and not yet well understood, giving rise to emerging and evolving attack vectors. The Kubernetes attack matrix provides an important, starting foundation for the Kubernetes community and cloud-native ecosystem to effectively protect Kubernetes, and the applications that run on it, from common tactics and techniques that might be used by adversaries.

The rich set of security controls made available by both Kubernetes and cloud provider platforms lend themselves to a set of security best practices that organizations can implement to mitigate the threats described in the Kubernetes attack matrix. This aligns with established frameworks for security such as the shared responsibility model for cloud environments.

To comprehensively mitigate the threats listed in the attack matrix requires a security approach that goes beyond being container-centric. It requires security that is Kubernetes-native: security that is fully architected to integrate with and leverage native security controls within Kubernetes itself.

New Kubernetes attack vectors will continue to be discovered over time alongside the growth of the platform, and organizations should be prepared to expand their Kubernetes security efforts over time to accommodate these developments. StackRox's Kubernetes-native security is designed to address new threats to Kubernetes applications and infrastructure as they emerge.

As the industry's only Kubernetes-native security solution, StackRox's Kubernetes-native security delivers several unique benefits to organizations building and running containerized applications today.

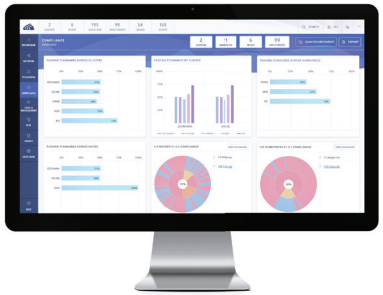
- It provides increased protection by eliminating blind spots as well as discovering critical vulnerabilities and misconfigurations unique to Kubernetes.
- It saves time and costs by shortening the learning curve for teams and accelerating investigation and remediation with valuable context from Kubernetes.
- It minimizes overall operational risk by leveraging Kubernetes for scalable enforcement functions while eliminating operational complexity that arises from inconsistent configurations and user error.
- It gives development, operations, and security teams a single, standardized platform for all key security use cases across the cloud-native application life cycle.

With StackRox, organizations can fully secure their Kubernetes environments anywhere, in production, at scale.

StackRox

About StackRox

The StackRox Kubernetes Security Platform is the industry's first and only Kubernetes-native container security platform. It enables security and DevOps teams to work together to protect containerized applications from build to deploy to runtime. It integrates with security and DevOps toolchains to help teams operationalize security and compliance for cloud-native applications. Organizations of all sizes, including cloud-native startups, large enterprises, and government agencies, rely on StackRox to protect their most innovative applications. For more information, visit www.stackrox.com.



Ready to see StackRox in action?

Get a personalized demo tailored for your business, environment, and needs.

REQUEST DEMO