Security Audit Report

Files Scanned: ["sample-k8s-large-vuln.yaml"]

Date Generated: 01-10-2025

Kubernetes Security Report

Technical Summary

This report summarizes the findings from the security scans conducted using Kubescape and Trivy on the provided Kubernetes manifest. The analysis revealed a total of **32 critical/high issues** across various categories, including:

- **RBAC Misconfigurations**: 5 issues
- **Pod Security Violations**: 10 issues
- **Secrets Management**: 3 issues
- **Network Policies**: 2 issues
- **Resource Management**: 4 issues
- **General Misconfigurations**: 8 issues

The main risk categories identified are RBAC, Pod Security, and Secrets Management, which pose significant threats to the integrity and confidentiality of the Kubernetes environment.

Prioritized Findings & Investigation Playbook

Finding 1: Privileged Container Detected

- **Severity**: Critical
- **Technical Explanation**: The `privileged-pod` runs a container with `privileged: true`, allowing it to access all devices on the host and perform any operation. This can lead to a complete host compromise.
- **Probing Questions & Indicative Responses**:
  - **Q1**: Is the `privileged` flag necessary for this container?
    - **Positive Response**: "No, it can run without elevated privileges."
    - **Negative Response**: "Yes, it needs full access."
  - **Q2**: Are there any security controls in place to monitor privileged containers?
    - **Positive Response**: "Yes, we have monitoring in place."
    - **Negative Response**: "No, we do not monitor them."
- **OWASP & Compliance Mapping**: OWASP Kubernetes Top Ten - K8s-003 (Excessive Privileges), CIS Kubernetes Benchmark - 5.1
- **Remediation Guidance**: Remove the `privileged: true` setting from the container's security context.

Finding 2: HostPath Volume Mounted with Docker Socket

- **Severity**: Critical
- **Technical Explanation**: Mounting the Docker socket (`/var/run/docker.sock`) allows the container to control the Docker daemon, leading to potential host access and privilege

escalation.

- **Probing Questions & Indicative Responses**:
  - **Q1**: Is there a specific need to mount the Docker socket?
    - **Positive Response**: "No, it's not required."
    - **Negative Response**: "Yes, we need it for container management."
  - **Q2**: Are there alternative methods to achieve the same functionality without mounting the socket?
    - **Positive Response**: "Yes, we can use Kubernetes APIs."
    - **Negative Response**: "No, this is the only way."
- **OWASP & Compliance Mapping**: OWASP Kubernetes Top Ten - K8s-003 (Excessive Privileges), CIS Kubernetes Benchmark - 5.3
- **Remediation Guidance**: Remove the volume mount for `/var/run/docker.sock`.

Finding 3: ClusterRole with Cluster-Admin Privileges

- **Severity**: Critical
- **Technical Explanation**: The `omnipotent-role` grants unrestricted access to all resources in the cluster, which can lead to severe security breaches if misused.
- **Probing Questions & Indicative Responses**:
  - **Q1**: Is this role necessary for the service account?
    - **Positive Response**: "No, it can be scoped down."
    - **Negative Response**: "Yes, it needs full access."
  - **Q2**: Are there any audits performed on the usage of this role?
    - **Positive Response**: "Yes, we audit role usage."
    - **Negative Response**: "No, we do not audit."
- **OWASP & Compliance Mapping**: OWASP Kubernetes Top Ten - K8s-002 (RBAC Misconfigurations), CIS Kubernetes Benchmark - 1.1
- **Remediation Guidance**: Limit the role to specific resources and actions necessary for the service account.

Finding 4: Plaintext Secret Detected

- **Severity**: High
- **Technical Explanation**: Storing sensitive information like database credentials in plaintext exposes them to anyone with access to the Kubernetes API or manifests.
- **Probing Questions & Indicative Responses**:
  - **Q1**: Are secrets encrypted at rest in your cluster?
    - **Positive Response**: "Yes, we use encryption."
    - **Negative Response**: "No, they are stored as plaintext."
  - **Q2**: Is there a policy for managing secrets securely?
    - **Positive Response**: "Yes, we have a policy."
    - **Negative Response**: "No, we do not have a policy."
- **OWASP & Compliance Mapping**: OWASP Kubernetes Top Ten - K8s-004 (Secrets Management), CIS Kubernetes Benchmark - 5.4

Management), CIS Kubernetes Benchmark - 5.4
- **Remediation Guidance**: Store secrets using Kubernetes Secrets with base64 encoding and enable encryption at rest.

Finding 5: Insecure Image Tag Used

- **Severity**: High
- **Technical Explanation**: The `nginx:1.17` image is outdated and may contain known vulnerabilities. Using specific tags instead of `latest` or outdated versions can lead to security risks.
- **Probing Questions & Indicative Responses**:
  - **Q1**: Are images regularly scanned for vulnerabilities?
    - **Positive Response**: "Yes, we scan images regularly."
    - **Negative Response**: "No, we do not scan."
  - **Q2**: Is there a policy for using only trusted images?
    - **Positive Response**: "Yes, we have a policy."
    - **Negative Response**: "No, we do not have a policy."
- **OWASP & Compliance Mapping**: OWASP Kubernetes Top Ten - K8s-001 (Vulnerable Images), CIS Kubernetes Benchmark - 2.1
- **Remediation Guidance**: Update to a more recent and secure image version.

Cluster/Namespace Impact Overview

The `test-space` namespace is heavily impacted by the identified issues, with all critical/high findings concentrated in this namespace. The following resources are particularly vulnerable:

- Pods: `privileged-pod`, `vulnerable-deployment`, `naughty-cron`
- RBAC: `omnipotent-role`, `omnipotent-binding`
- Secrets: `db-credentials-plain`

Manifest Content Analysis

Upon manual review of the manifest, several logical flaws were identified that automated scanners might miss:

1. **Overly Permissive Network Policies**: The `allow-all-policy` allows all ingress and egress traffic, which can lead to lateral movement within the cluster.
2. **Insecure Ingress Routing**: The `insecure-ingress` lacks TLS, exposing traffic to potential interception.
3. **Secrets Exposure**: The `ConfigMap` contains sensitive information (API tokens) that should not be stored in plaintext.
4. **Resource Quota Misconfiguration**: The `tiny-quota` is set too low, risking cluster starvation during peak loads.

Next Steps & Ownership

1. **Remediation of Critical Findings**:
    - **Responsible Team**: DevOps
    - **Action**: Implement the remediation guidance for all critical findings.
2. **Review and Update Security Policies**:
    - **Responsible Team**: AppSec
    - **Action**: Establish and enforce policies for secrets management and image usage.
3. **Conduct Regular Security Audits**:
    - **Responsible Team**: SRE
    - **Action**: Schedule regular audits of RBAC roles and permissions.
4. **Implement Network Policies**:
    - **Responsible Team**: Network Engineering
    - **Action**: Define and enforce restrictive network policies to limit traffic.