Vera C. Rubin Observatory
Rubin Observatory Project Office

Hardening Kubernetes Workload

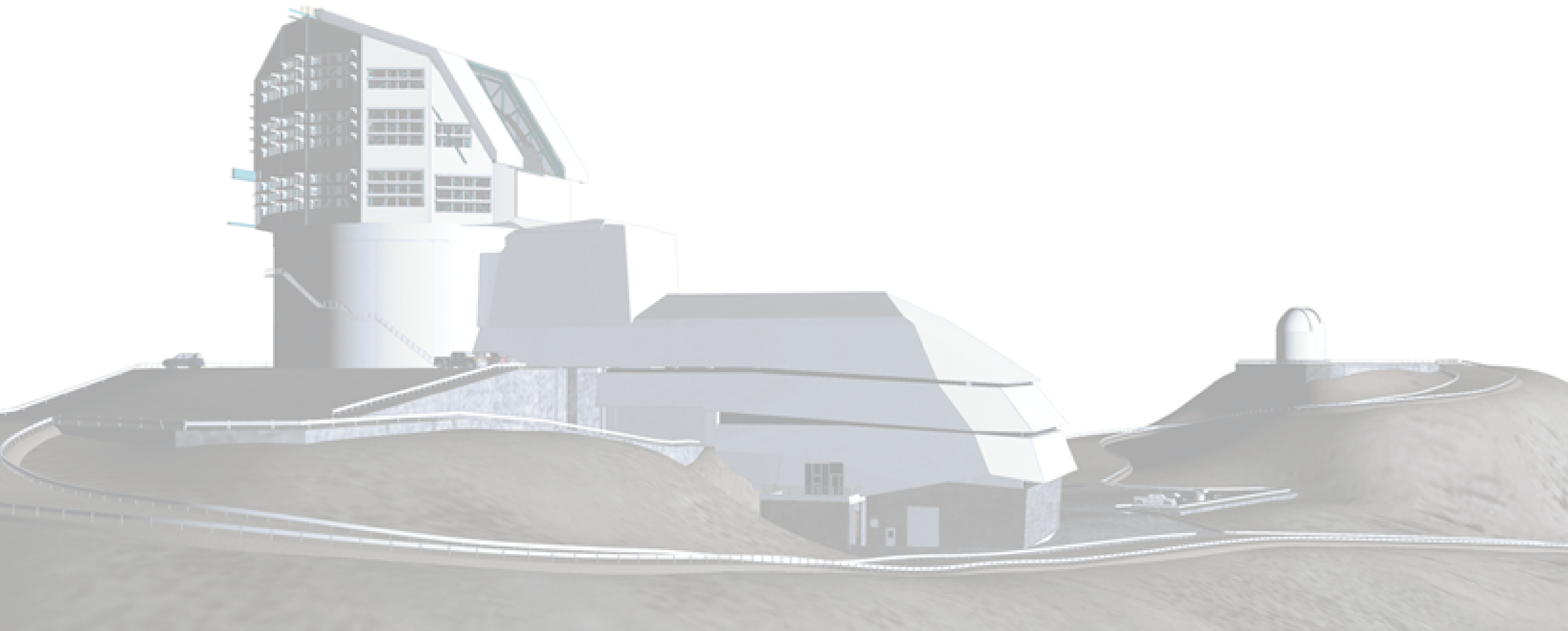
Gonzalo Seriche

ITTN-079

Latest Revision: 2024-12-19

Draft

DRAFT



Abstract

This technical note provides comprehensive guidance for implementing security controls and hardening measures for Kubernetes workloads based on CIS Benchmark recommendations. It covers pod security policies, network security, container image security, and compliance monitoring for on-premise Kubernetes deployments.

Draft

Change Record

Version	Date	Description	Owner name
1	2024-09-13	Initial release.	Gonzalo Seriche
1	2024-12-19	Add primary best practices and security guidelines	Gonzalo Seriche

Document source location: `https://github.com/lsst-it/ittn-079`

Draft

Draft

Contents

Draft

Hardening Kubernetes Workload

1 Introduction

This document outlines the security measures, implementation guidelines, and best practices for hardening Kubernetes workloads in accordance with the Center for Internet Security (CIS) Benchmark. The guidance provided here is specifically tailored for on-premise Kubernetes deployments and focuses on establishing a robust security posture while maintaining operational efficiency.

2 Pod Security Standards

2.1 Pod Security Context Implementation

Pod security context defines the privilege and access control settings that are essential for maintaining a secure Kubernetes environment. The following configurations establish the foundation for pod-level security:

Draft

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: secure-pod
spec:
  securityContext:
    runAsNonRoot: true
    runAsUser: 1000
    fsGroup: 3000
    seccompProfile:
      type: RuntimeDefault
```

The security context implementation enforces several critical security controls:

- Non-root execution prevents privilege escalation scenarios

- Specific UID assignments ensure precise access control
- Filesystem group configurations maintain proper data access boundaries
- SecComp profiles limit available system calls to reduce the attack surface

2.2 Container Level Security Controls

Container security context configurations provide granular control over container execution parameters:

```yaml
spec:
  containers:
  - name: secured-container
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop: ["ALL"]
      readOnlyRootFilesystem: true
      privileged: false
```

These settings establish a defensive posture through:

- Prevention of privilege escalation attempts
- Removal of unnecessary Linux capabilities
- Implementation of read-only root filesystems
- Explicit prohibition of privileged execution modes

3 Network Security Framework

3.1 Default Network Policies

Implementation of a default deny policy establishes a zero-trust networking baseline:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

3.2 Granular Access Controls

Specific allow rules should be implemented for required communications:

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-specific
spec:
  podSelector:
    matchLabels:
      app: web
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 80
```

4 Resource Management and Quotas

Draft

4.1 Resource Quota Implementation

Resource quotas prevent resource exhaustion and ensure fair resource allocation:

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
spec:
  hard:
    requests.cpu: "4"
    requests.memory: 8Gi
    limits.cpu: "8"
    limits.memory: 16Gi
```

4.2 Default Resource Constraints

Limit ranges establish default resource boundaries:

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: resource-constraints
spec:
  limits:
  - default:
      cpu: 500m
      memory: 512Mi
    defaultRequest:
      cpu: 200m
      memory: 256Mi
    type: Container
```

Dra

5 Monitoring and Compliance Framework

5.1 Audit Logging Configuration

Comprehensive audit logging enables security monitoring and compliance verification:

```yaml
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
- level: Metadata
  resources:
  - group: ""
    resources: ["pods", "services"]
- level: RequestResponse
  resources:
  - group: "apps"
    resources: ["deployments"]
```

5.2 Compliance Monitoring Implementation

The compliance monitoring framework should encompass:

- Automated security scanning tools integration
- Continuous compliance status monitoring
- Policy enforcement automation
- Regular CIS Benchmark assessment procedures

6 Security Best Practices

6.1 Image Security Controls

Image security measures should include:

- Mandatory image signing and verification
- Regular vulnerability scanning
- Base image update procedures
- Registry access control implementation

6.2 Secret Management

Implement robust secret management practices:

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: application-secrets
  annotations:
    vault.security.banzaicloud.io/vault-role: "app-role"
type: Opaque
data:
  APP_SECRET: <base64-encoded-secret>
```

Draft

7 Implementation Guidelines

7.1 Deployment Process

The security control implementation process should follow these phases:

1. Security posture assessment
2. Gap analysis against CIS Benchmark
3. Control implementation prioritization
4. Gradual security measure deployment

5. Validation and testing
6. Continuous monitoring establishment

7.2 Maintenance Procedures

Establish regular maintenance procedures for:

- Security patch management
- Configuration updates
- Policy refinement
- Compliance verification

8 CIS Benchmark Compliance Testing

8.1 Kube-bench Implementation

Kube-bench provides automated testing for Kubernetes CIS Benchmark compliance. The implementation varies based on Kubernetes versions and deployment methods.

8.1.1 Version Compatibility

Kube-bench supports multiple Kubernetes releases with specific test definitions:

```
# Job specification for Kubernetes 1.24+
apiVersion: batch/v1
kind: Job
metadata:
  name: kube-bench
spec:
  template:
    spec:
```

Draft

```
      hostPID: true
      containers:
        - name: kube-bench
          image: aquasec/kube-bench:v0.6.15
          command: ["kube-bench", "run", "--targets", "master,node"]
          volumeMounts:
            - name: var-lib-kubelet
              mountPath: /var/lib/kubelet
            - name: etc-systemd
              mountPath: /etc/systemd
            - name: etc-kubernetes
              mountPath: /etc/kubernetes
      restartPolicy: Never
      volumes:
        - name: var-lib-kubelet
          hostPath:
            path: "/var/lib/kubelet"
        - name: etc-systemd
          hostPath:
            path: "/etc/systemd"
        - name: etc-kubernetes
          hostPath:
            path: "/etc/kubernetes"
```

8.1.2 Automated Scheduling

Implement regular compliance testing through CronJobs:

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: kube-bench-schedule
spec:
  schedule: "0 1 * * *"  # Run daily at 1 AM
  jobTemplate:
    spec:
      template:
        spec:
```

```
      hostPID: true
      containers:
        - name: kube-bench
          image: aquasec/kube-bench:v0.6.15
          command:
            - "kube-bench"
            - "run"
            - "--targets"
            - "master,node"
            - "--outputfile"
            - "/reports/compliance.json"
          volumeMounts:
            - name: var-lib-kubelet
              mountPath: /var/lib/kubelet
            - name: reports
              mountPath: /reports
      volumes:
        - name: var-lib-kubelet
          hostPath:
            path: "/var/lib/kubelet"
        - name: reports
          persistentVolumeClaim:
            claimName: compliance-reports-pvc
      restartPolicy: Never
```

8.2 Version-Specific Configurations

Different Kubernetes versions require specific considerations:

For Kubernetes 1.24 and newer:

- Use kube-bench version 0.6.x
- Enable container runtime socket mounting
- Configure specific test suites for newer security features

For Kubernetes 1.23 and older:

- Use compatible kube-bench versions (0.5.x)
- Adjust for legacy API versions
- Include deprecated security controls testing

8.3 Results Processing

Implement automated results processing:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: compliance-processor
data:
  process.sh: |
    #!/bin/bash
    REPORT_PATH="/reports/compliance.json"
    if [ -f "$REPORT_PATH" ]; then
      # Process results
      jq -r '.Controls[] | select(.status=="FAIL")' "$REPORT_PATH" > failures.json
      # Alert if necessary
      if [ -s failures.json ]; then
        # Send alerts
        curl -X POST ${ALERT_ENDPOINT} -d @failures.json
      fi
    fi
```

9 Compliance Monitoring and Alerting Framework

9.1 Advanced Results Processing

The compliance results processing system implements a comprehensive pipeline for analyzing kube-bench outputs and generating actionable insights. The system processes both JSON and YAML outputs, correlates findings across multiple clusters, and maintains a historical compliance database.

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: compliance-processor-config
data:
  config.yaml: |
    processing:
      output_format: json
      retention_days: 90
      minimum_severity: MEDIUM
      excluded_checks:
        - 1.2.1  # Document exclusion rationale
        - 2.1.3  # Alternative control implemented
    alerting:
      threshold_critical: 85
      threshold_warning: 95
      notification_channels:
        - slack
        - email
        - pagerduty
    reporting:
      format: html
      schedule: "0 8 * * 1"  # Weekly reports
      recipients: ["security-team@org.com"]
```

9.2 Monitoring Integration

The monitoring framework integrates with existing observability platforms through a comprehensive metrics pipeline:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: compliance-metrics
spec:
  selector:
    matchLabels:
      app: kube-bench
  endpoints:
  - port: metrics
    interval: 30s
    path: /metrics
  - port: compliance
    interval: 5m
    path: /compliance
---
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: compliance-alerts
spec:
  groups:
  - name: compliance.rules
    rules:
    - alert: ComplianceScoreDegraded
      expr: compliance_score < 85
      for: 1h
      labels:
        severity: critical
      annotations:
        summary: Cluster compliance score below threshold
        description: Cluster {{ $labels.cluster }} compliance score is {{ $value }}
    - alert: HighSeverityFinding
      expr: compliance_findings{severity="HIGH"} > 0
      for: 5m
      labels:
        severity: critical
```

9.3 Advanced Alerting Configuration

The alerting system implements a sophisticated notification framework with different severity levels and escalation paths:

```yaml
apiVersion: notification.toolkit.fluxcd.io/v1beta1
kind: Alert
metadata:
  name: compliance-alerts
spec:
  eventSeverity: info
  eventSources:
    - kind: Kustomization
      name: '*'
  providerRef:
    name: slack
---
apiVersion: notification.toolkit.fluxcd.io/v1beta1
kind: Provider
metadata:
  name: slack
spec:
  type: slack
  channel: security-alerts
  address: https://hooks.slack.com/services/YOUR-WEBHOOK-URL
  secretRef:
    name: slack-url
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: alert-templates
data:
  critical.tmpl: |
    *Critical Compliance Alert*
    Cluster: {{ .Cluster }}
    Check: {{ .CheckID }}
    Description: {{ .Description }}
    Remediation: {{ .Remediation }}
  warning.tmpl: |
    Warning: Compliance Issue Detected
    Cluster: {{ .Cluster }}
    Details: {{ .Details }}
```

9.4 Compliance Reporting and Analytics

The reporting system generates comprehensive compliance analytics and trend analysis:

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: compliance-report-generator
spec:
  schedule: "0 1 * * 1"  # Weekly on Monday at 1 AM
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: report-generator
            image: compliance-reporter:v1
            env:
            - name: REPORT_PERIOD
              value: "7d"
            - name: INCLUDE_TRENDS
              value: "true"
            - name: REPORT_FORMAT
              value: "html,pdf"
            volumeMounts:
            - name: report-output
              mountPath: /reports
          volumes:
          - name: report-output
            persistentVolumeClaim:
              claimName: report-storage
```

A References

B Acronyms

Acronym	Description
API	Application Programming Interface
CIS	Computer Infrastructure Support
ITTN	IT Technote
JSON	JavaScript Object Notation
PMO	Project Management Office
TCP	Transmission Control Protocol
UID	User Identifier
URL	Universal Resource Locator
YAML	Yet Another Markup Language

Draft