Security Audit Report

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

Date Generated: 01-10-2025

Remediation Report for Kubernetes Manifest

Action Summary

- **Namespace**: Remove or restrict `test-space` namespace usage.
- **Secrets**: Replace plaintext secrets in `db-credentials-plain` with base64 encoded values.
- **ConfigMap**: Remove sensitive data from `app-config`.
- **ServiceAccount**: Disable `automountServiceAccountToken` for `risky-sa`.
- **ClusterRole**: Remove `omniscient-role` or restrict its permissions.
- **ClusterRoleBinding**: Remove `omnipotent-binding` or bind to a less privileged role.
- `privileged-pod`
- **Deployment**: Update `vulnerable-deployment` to use a secure image tag and add resource limits.
- **DaemonSet**: Remove `hostPath` volume from `log-shipper`.
- **StatefulSet**: Replace `emptyDir` with a durable storage solution.
- **Service**: Change `NodePort` to `ClusterIP` or `LoadBalancer` with restricted access.
- **Ingress**: Enable TLS for `insecure-ingress`.
- **PodDisruptionBudget**: Increase `minAvailable` for `brittle-pdb`.
- **HorizontalPodAutoscaler**: Set a more appropriate `minReplicas`.
- **Role**: Restrict permissions in `pod-access-role`.
- **RoleBinding**: Avoid binding `default` service account to `pod-access-role`.
- **CronJob**: Remove `hostPath` mount and run as a non-root user.
- **Pod**: Change `imagePullPolicy` from `Never` to `IfNotPresent` in `bad-pull-policy`.
- **ResourceQuota**: Ensure pods have resource requests defined.
- **LimitRange**: Set stricter limits in `permissive-limits`.
- **NetworkPolicy**: Implement restrictive ingress/egress rules.
- **Pod**: Remove `hostIPC` from `ipc-pod`.
- **Deprecated API**: Update any deprecated API usages.

Remediation Playbook

1. Namespace: `test-space`

- **What is the issue?**: The namespace is configured for insecure testing.
- **Why does it matter?**: It can lead to accidental deployment of insecure workloads in production.
- **How to fix it**: Remove or restrict the namespace.

```
kubectl delete namespace test-space
```

- **Validation Step**:

```
kubectl get namespaces
```

2. Plaintext Secret: `db-credentials-plain`

- **What is the issue?**: Secrets are stored in plaintext.
- **Why does it matter?**: Exposes sensitive data to anyone with access to the manifest.
- **How to fix it**: Encode secrets in base64.

```
apiVersion: v1
kind: Secret
metadata:
  name: db-credentials-plain
  namespace: test-space
type: Opaque
data:
  username: YWRtaW4=  # base64 encoded
  password: Q29ycmVjdEhvc2VCYXR0ZXJ5U3RhcGxlMTIzIQ==  # base64 encoded
```

- **Validation Step**:

```
kubectl get secret db-credentials-plain -n test-space -o yaml
```

3. ConfigMap: `app-config`

- **What is the issue?**: Sensitive information is stored in a ConfigMap.
- **Why does it matter?**: Exposes sensitive data to anyone with access to the ConfigMap.
- **How to fix it**: Remove sensitive data.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
  namespace: test-space
data:
  REDIS_URL: "redis://10.0.0.5:6379"
```

- **Validation Step**:

```
kubectl get configmap app-config -n test-space -o yaml
```

4. ServiceAccount: `risky-sa`

- **What is the issue?**: Token automount is enabled.
- **Why does it matter?**: Increases risk of privilege escalation.
- **How to fix it**: Disable token automount.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: risky-sa
  namespace: test-space
automountServiceAccountToken: false
```

- **Validation Step**:

```
kubectl get serviceaccount risky-sa -n test-space -o yaml
```

5. ClusterRole: `omniscient-role`

- **What is the issue?**: Grants excessive permissions.
- **Why does it matter?**: Increases risk of unauthorized access.
- **How to fix it**: Remove or restrict the role.

```
kubectl delete clusterrole omnipotent-role
```

- **Validation Step**:

```
kubectl get clusterroles
```

6. ClusterRoleBinding: `omnipotent-binding`

- **What is the issue?**: Binds a service account to a cluster-admin role.
- **Why does it matter?**: Grants excessive permissions to a namespaced service account.
- **How to fix it**: Remove the binding.

```
kubectl delete clusterrolebinding omnipotent-binding
```

- **Validation Step**:

```
kubectl get clusterrolebindings
```

7. Pod: `privileged-pod`

- **What is the issue?**: Pod runs with privileged access and hostPath mounts.
- **Why does it matter?**: Exposes the host to potential security risks.
- **How to fix it**: Remove privileged settings and hostPath mounts.

```
apiVersion: v1
kind: Pod
metadata:
  name: privileged-pod
  namespace: test-space
spec:
  containers:
    - name: privileged-sidecar
      image: alpine:3.12
      securityContext:
        privileged: false
        allowPrivilegeEscalation: false
        runAsUser: 1000  # non-root user
```

- **Validation Step**:

```
kubectl describe pod privileged-pod -n test-space
```

8. Deployment: `vulnerable-deployment`

- **What is the issue?**: Uses an insecure image tag and lacks resource limits.
- **Why does it matter?**: Increases risk of running outdated or vulnerable software.
- **How to fix it**: Update the image and add resource limits.

```
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: vulnerable-deployment
  namespace: test-space
spec:
  template:
    spec:
      containers:
        - name: web
          image: nginx:latest  # updated image
          resources:
            requests:
              memory: "128Mi"
              cpu: "500m"
            limits:
              memory: "256Mi"
              cpu: "1"
```

- **Validation Step**:

```
kubectl get deployment vulnerable-deployment -n test-space -o yaml
```

9. DaemonSet: `log-shipper`

- **What is the issue?**: Uses hostPath volume.
- **Why does it matter?**: Can expose host filesystem to containers.
- **How to fix it**: Remove the hostPath volume.

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: log-shipper
  namespace: test-space
spec:
  template:
    spec:
      containers:
        - name: shipper
          image: busybox:1.28
```

- **Validation Step**:

```
```

```
kubectl describe daemonset log-shipper -n test-space
```

10. StatefulSet: `bogus-db`

- **What is the issue?**: Uses emptyDir for storage.
- **Why does it matter?**: Data is lost on pod eviction or restart.
- **How to fix it**: Use a persistent volume claim instead.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: bogus-db
  namespace: test-space
spec:
  template:
    spec:
      containers:
        - name: fake-db
          image: postgres:latest  # updated image
          volumeMounts:
            - name: data
              mountPath: /var/lib/postgresql/data
      volumes:
        - name: data
          persistentVolumeClaim:
            claimName: postgres-pvc  # ensure PVC is created
```

- **Validation Step**:

```
kubectl get statefulset bogus-db -n test-space -o yaml
```

11. Service: `external-web`

- **What is the issue?**: Exposes pods via NodePort.
- **Why does it matter?**: Can expose services to the internet unnecessarily.
- **How to fix it**: Change to `ClusterIP`.

```
apiVersion: v1
kind: Service
metadata:
  name: external-web
  namespace: test-space
```

```
  namespace: test-space
spec:
  type: ClusterIP
```

- **Validation Step**:

```
kubectl get service external-web -n test-space -o yaml
```

12. Ingress: `insecure-ingress`

- **What is the issue?**: Missing TLS configuration.
- **Why does it matter?**: Exposes traffic to potential interception.
- **How to fix it**: Enable TLS.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: insecure-ingress
  namespace: test-space
spec:
  tls:
    - hosts:
        - insecure.example.test
      secretName: tls-secret  # Ensure this secret exists
  rules:
    - host: insecure.example.test
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: external-web
                port:
                  number: 80
```

- **Validation Step**:

```
kubectl describe ingress insecure-ingress -n test-space
```

13. PodDisruptionBudget: `brittle-pdb`

- **What is the issue?**: Low `minAvailable` setting.
- **Why does it matter?**: Can lead to service disruption during maintenance.
- **How to fix it**: Increase `minAvailable`.

```
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: brittle-pdb
  namespace: test-space
spec:
  minAvailable: 2  # increase to ensure availability
```

- **Validation Step**:

```
kubectl get poddisruptionbudget brittle-pdb -n test-space -o yaml
```

14. HorizontalPodAutoscaler: `dumb-hpa`

- **What is the issue?**: Permissive `minReplicas`.
- **Why does it matter?**: Can lead to resource exhaustion.
- **How to fix it**: Set a more appropriate `minReplicas`.

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: dumb-hpa
  namespace: test-space
spec:
  minReplicas: 2  # set to a more appropriate value
```

- **Validation Step**:

```
kubectl get hpa dumb-hpa -n test-space -o yaml
```

15. Role: `pod-access-role`

- **What is the issue?**: Too permissive permissions.
- **Why does it matter?**: Increases risk of unauthorized access.
- **How to fix it**: Restrict permissions.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: test-space
  name: pod-access-role
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch"]
```

- **Validation Step**:

```
kubectl get role pod-access-role -n test-space -o yaml
```

16. RoleBinding: `default-pod-access`

- **What is the issue?**: Binds the default service account to a permissive role.
- **Why does it matter?**: Increases risk of unauthorized access.
- **How to fix it**: Remove the binding.

```
kubectl delete rolebinding default-pod-access -n test-space
```

- **Validation Step**:

```
kubectl get rolebindings -n test-space
```

17. CronJob: `naughty-cron`

- **What is the issue?**: Runs as root and mounts hostPath.
- **Why does it matter?**: Increases risk of privilege escalation.
- **How to fix it**: Run as a non-root user and remove hostPath.

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: naughty-cron
  namespace: test-space
spec:
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: runner
              image: alpine:3.12
              securityContext:
                runAsUser: 1000  # non-root user
          restartPolicy: OnFailure
```

- **Validation Step**:

```
kubectl describe cronjob naughty-cron -n test-space
```

18. Pod: `bad-pull-policy`

- **What is the issue?**: Uses `imagePullPolicy: Never`.
- **Why does it matter?**: Prevents pulling updates and can fail if the image is not present locally.
- **How to fix it**: Change to `IfNotPresent`.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: bad-pull-policy
  namespace: test-space
spec:
  containers:
    - name: remote
      image: myregistry.example.com/some/image:latest
      imagePullPolicy: IfNotPresent
```

- **Validation Step**:

```
kubectl describe pod bad-pull-policy -n test-space
```

19. ResourceQuota: `tiny-quota`

- **What is the issue?**: Pods lack resource requests.
- **Why does it matter?**: Can lead to cluster resource starvation.
- **How to fix it**: Ensure pods have resource requests defined.

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: tiny-quota
  namespace: test-space
spec:
  hard:
    requests.cpu: "500m"
    requests.memory: "512Mi"
```

- **Validation Step**:

```
kubectl get resourcequota tiny-quota -n test-space -o yaml
```

20. LimitRange: `permissive-limits`

- **What is the issue?**: Missing limits or permissive defaults.
- **Why does it matter?**: Can lead to resource exhaustion.
- **How to fix it**: Set stricter limits.

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: permissive-limits
  namespace: test-space
spec:
  limits:
    - defaultRequest:
        cpu: 100m
        memory: 128Mi
      default:
        cpu: 200m
        memory: 256Mi
      type: Container
```

- **Validation Step**:

```
kubectl get limitrange permissive-limits -n test-space -o yaml
```

21. NetworkPolicy: `allow-all-policy`

- **What is the issue?**: Allows all ingress/egress traffic.
- **Why does it matter?**: No network segmentation increases risk.
- **How to fix it**: Implement restrictive rules.

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-all-policy
  namespace: test-space
spec:
  podSelector: {}
```