CIS Oracle Cloud Infrastructure Container Engine for Kubernetes(OKE) Benchmark

v1.8.0 - 11-25-2025

Terms of Use

Please see the below link for our current terms of use:

https://www.cisecurity.org/cis-securesuite/cis-securesuite-membership-terms-of-use/

For information on referencing and/or citing CIS Benchmarks in 3rd party documentation (including using portions of Benchmark Recommendations) please contact CIS Legal (legalnotices@cisecurity.org) and request guidance on copyright usage.

NOTE: It is ***NEVER*** acceptable to host a CIS Benchmark in ***ANY*** format (PDF, etc.) on a 3rd party (non-CIS owned) site.

Table of Contents

Overview

All CIS Benchmarks™ (Benchmarks) focus on technical configuration settings used to maintain and/or increase the security of the addressed technology, and they should be used in **conjunction** with other essential cyber hygiene tasks like:

- Monitoring the base operating system and applications for vulnerabilities and quickly updating with the latest security patches.
- End-point protection (Antivirus software, Endpoint Detection and Response (EDR), etc.).
- Logging and monitoring user and system activity.

In the end, the Benchmarks are designed to be a key **component** of a comprehensive cybersecurity program.

Important Usage Information

All Benchmarks are available free for non-commercial use from the CIS Website. They can be used to manually assess and remediate systems and applications. In lieu of manual assessment and remediation, there are several tools available to assist with assessment:

- CIS Configuration Assessment Tool (CIS-CAT® Pro Assessor)
- CIS Benchmarks™ Certified 3rd Party Tooling

These tools make the hardening process much more scalable for large numbers of systems and applications.

NOTE: Some tooling focuses only on the Benchmark Recommendations that can be fully automated (skipping ones marked **Manual**). It is important that ***ALL*** Recommendations (**Automated** and **Manual**) be addressed since all are important for properly securing systems and are typically in scope for audits.

Key Stakeholders

Cybersecurity is a collaborative effort, and cross functional cooperation is imperative within an organization to discuss, test, and deploy Benchmarks in an effective and efficient way. The Benchmarks are developed to be best practice configuration guidelines applicable to a wide range of use cases. In some organizations, exceptions to specific Recommendations will be needed, and this team should work to prioritize the problematic Recommendations based on several factors like risk, time, cost, and labor. These exceptions should be properly categorized and documented for auditing purposes.

Apply the Correct Version of a Benchmark

Benchmarks are developed and tested for a specific set of products and versions and applying an incorrect Benchmark to a system can cause the resulting pass/fail score to be incorrect. This is due to the assessment of settings that do not apply to the target systems. To assure the correct Benchmark is being assessed:

- **Deploy the Benchmark applicable to the way settings are managed in the environment:** An example of this is the Microsoft Windows family of Benchmarks, which have separate Benchmarks for Group Policy, Intune, and Stand-alone systems based upon how system management is deployed. Applying the wrong Benchmark in this case will give invalid results.

- **Use the most recent version of a Benchmark**: This is true for all Benchmarks, but especially true for cloud technologies. Cloud technologies change frequently and using an older version of a Benchmark may have invalid methods for auditing and remediation.

Exceptions

The guidance items in the Benchmarks are called recommendations and not requirements, and exceptions to some of them are expected and acceptable. The Benchmarks strive to be a secure baseline, or starting point, for a specific technology, with known issues identified during Benchmark development are documented in the Impact section of each Recommendation. In addition, organizational, system specific requirements, or local site policy may require changes as well, or an exception to a Recommendation or group of Recommendations (e.g. A Benchmark could Recommend that a Web server not be installed on the system, but if a system's primary purpose is to function as a Webserver, there should be a documented exception to this Recommendation for that specific server).

In the end, exceptions to some Benchmark Recommendations are common and acceptable, and should be handled as follows:

- The reasons for the exception should be reviewed cross-functionally and be well documented for audit purposes.
- A plan should be developed for mitigating, or eliminating, the exception in the future, if applicable.
- If the organization decides to accept the risk of this exception (not work toward mitigation or elimination), this should be documented for audit purposes.

It is the responsibility of the organization to determine their overall security policy, and which settings are applicable to their unique needs based on the overall risk profile for the organization.

Remediation

CIS has developed Build Kits for many technologies to assist in the automation of hardening systems. Build Kits are designed to correspond to Benchmark's "Remediation" section, which provides the manual remediation steps necessary to make that Recommendation compliant to the Benchmark.

When remediating systems (changing configuration settings on deployed systems as per the Benchmark's Recommendations), please approach this with caution and test thoroughly.

The following is a reasonable remediation approach to follow:

- CIS Build Kits, or internally developed remediation methods should never be applied to production systems without proper testing.
- Proper testing consists of the following:
  - Understand the configuration (including installed applications) of the targeted systems. Various parts of the organization may need different configurations (e.g., software developers vs standard office workers).
  - Read the Impact section of the given Recommendation to help determine if there might be an issue with the targeted systems.
  - Test the configuration changes with representative lab system(s). If issues arise during testing, they can be resolved prior to deploying to any production systems.
  - When testing is complete, initially deploy to a small sub-set of production systems and monitor closely for issues. If there are issues, they can be resolved prior to deploying more broadly.
  - When the initial deployment above is completes successfully, iteratively deploy to additional systems and monitor closely for issues. Repeat this process until the full deployment is complete.

Summary

Using the Benchmarks Certified tools, working as a team with key stakeholders, being selective with exceptions, and being careful with remediation deployment, it is possible to harden large numbers of deployed systems in a cost effective, efficient, and safe manner.

NOTE: As previously stated, the PDF versions of the CIS Benchmarks™ are available for free, non-commercial use on the CIS Website. All other formats of the CIS Benchmarks™ (MS Word, Excel, and Build Kits) are available for CIS SecureSuite® members.

CIS-CAT® Pro is also available to CIS SecureSuite® members.

Target Technology Details

This document provides prescriptive guidance for running Oracle Kubernetes Engine (OKE) vNext following recommended security controls. This benchmark only includes controls which can be modified by an end user of OKE. For information on OKE's performance against the Kubernetes CIS benchmarks, for items which cannot be audited or modified, see the OKE documentation at https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

To obtain the latest version of this guide, please visit www.cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at support@cisecurity.org.

Intended Audience

This document is intended for cluster administrators, security specialists, auditors, and any personnel who plan to develop, deploy, assess, or secure solutions that incorporate Oracle Kubernetes Engine (OKE).

Consensus Guidance

This CIS Benchmark™ was created using a consensus review process comprised of a global community of subject matter experts. The process combines real world experience with data-based information to create technology specific guidance to assist users to secure their environments. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

Each CIS Benchmark undergoes two phases of consensus review. The first phase occurs during initial Benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the Benchmark. This discussion occurs until consensus has been reached on Benchmark recommendations. The second phase begins after the Benchmark has been published. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the Benchmark. If you are interested in participating in the consensus process, please visit https://workbench.cisecurity.org/.

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
`Monospace font`	Used for inline code, commands, UI/Menu selections or examples. Text should be interpreted exactly as presented.
`<Monospace font in brackets>`	Text set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to reference other relevant settings, CIS Benchmarks and/or Benchmark Communities. Also, used to denote the title of a book, article, or other publication.
Bold font	Additional information or caveats things like **Notes**, **Warnings**, or **Cautions** (usually just the word itself and the rest of the text normal).

Recommendation Definitions

The following defines the various components included in a CIS recommendation as applicable. If any of the components are not applicable it will be noted, or the component will not be included in the recommendation.

Title

Concise description for the recommendation's intended configuration.

Assessment Status

An assessment status is included for every recommendation. The assessment status indicates whether the given recommendation can be automated or requires manual steps to implement. Both statuses are equally important and are determined and supported as defined below:

Automated

Represents recommendations for which assessment of a technical control can be fully automated and validated to a pass/fail state. Recommendations will include the necessary information to implement automation.

Manual

Represents recommendations for which assessment of a technical control cannot be fully automated and requires all or some manual steps to validate that the configured state is set as expected. The expected state can vary depending on the environment.

Profile

A collection of recommendations for securing a technology or a supporting platform. Most benchmarks include at least a Level 1 and Level 2 Profile. Level 2 extends Level 1 recommendations and is not a standalone profile. The Profile Definitions section in the benchmark provides the definitions as they pertain to the recommendations included for the technology.

Description

Detailed information pertaining to the setting with which the recommendation is concerned. In some cases, the description will include the recommended value.

Rationale Statement

Detailed reasoning for the recommendation to provide the user a clear and concise understanding on the importance of the recommendation.

Impact Statement

Any security, functionality, or operational consequences that can result from following the recommendation.

Audit Procedure

Systematic instructions for determining if the target system complies with the recommendation.

Remediation Procedure

Systematic instructions for applying recommendations to the target system to bring it into compliance according to the recommendation.

Default Value

Default value for the given setting in this recommendation, if known. If not known, either not configured or not defined will be applied.

References

Additional documentation relative to the recommendation.

CIS Critical Security Controls® (CIS Controls®)

The mapping between a recommendation and the CIS Controls is organized by CIS Controls version, Safeguard, and Implementation Group (IG). The Benchmark in its entirety addresses the CIS Controls safeguards of (v7) “5.1 - Establish Secure Configurations” and (v8) '4.1 - Establish and Maintain a Secure Configuration Process” so individual recommendations will not be mapped to these safeguards.

Additional Information

Supplementary information that does not correspond to any other field but may be useful to the user.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

  Worker Node Profile

- **Level 2**

  Extends Level 1

Acknowledgements

This Benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Author/s
Gilson Melo
Adao Oliveira

Editor
Randall Mowen

Contributors
Mark Larinde
Logan Kleier
Adao Oliveira Junior KCNA, CKA, Oracle Corporation
Sherwood Zern
Josh Hammer
Yogesh Asalkar

Recommendations

1 Control Plane Components

Security is a shared responsibility between Oracle and the OKE customer. Under the Oracle Cloud Infrastructure Shared Security Model

Security of the cloud – Oracle is responsible for protecting the infrastructure that runs Oracle services in the Oracle Cloud. For OKE, Oracle is responsible for the Kubernetes control plane, which includes the control plane nodes and etcd database. Third-party auditors regularly test and verify the effectiveness of our security as part of the Oracle compliance programs.

Security in the cloud – Your responsibility includes the following areas:

- The security configuration of the data plane, including the configuration of the security groups that allow traffic to pass from the Oracle OKE control plane into the customer Virtual Cloud Network (VCN)
- The configuration of the worker nodes and the containers themselves
- The worker node guest operating system (including updates and security patches)
    - OKE follows the shared responsibility model for CVEs and security patches on managed node groups. Because managed nodes run the OKE-optimized OPIs, OKE is responsible for building patched versions of these OPIs when bugs or issues are reported and we are able to publish a fix. Customers are responsible for deploying these patched OPI versions to your managed node groups.
- Other associated application software:
    - Setting up and managing network controls, such as firewall rules
    - Managing platform-level identity and access management, either with or in addition to IAM
- The sensitivity of your data, your company's requirements, and applicable laws and regulations

Oracle is responsible for securing the control plane, though you might be able to configure certain options based on your requirements. Section 2 of this Benchmark addresses these configurations.

2 Control Plane Configuration

This section contains recommendations for Oracle OKE control plane configuration. Customers are able to configure logging for control plane in Oracle OKE.

2.1 Authentication and Authorization

2.1.1 Client certificate authentication should not be used for users (Manual)

Profile Applicability:

- Level 1

Description:

Kubernetes provides the option to use client certificates for user authentication. However as there is no way to revoke these certificates when a user leaves an organization or loses their credential, they are not suitable for this purpose.

It is not possible to fully disable client certificate use within a cluster as it is used for component to component authentication.

Rationale:

With any authentication mechanism the ability to revoke credentials if they are compromised or no longer required, is a key control. Kubernetes client certificate authentication does not allow for this due to a lack of support for certificate revocation.

Impact:

External mechanisms for authentication generally require additional software to be deployed.

Audit:

Review user access to the cluster and ensure that users are not making use of Kubernetes client certificate authentication.

You can verify the availability of client certificates in your OKE cluster.

Run the following command to verify the availability of client certificates in your OKE cluster:

```
kubectl get secrets --namespace kube-system
```

This command lists all the secrets in the kube-system namespace, which includes the client certificates used for authentication.

Look for secrets with names starting with oci- or oke-. These secrets contain the client certificates. If the command returns secrets with such names, it indicates that client certificates are available in your OKE cluster.

Remediation:

Alternative mechanisms provided by Kubernetes such as the use of OIDC should be implemented in place of client certificates.

You can remediate the availability of client certificates in your OKE cluster.

Default Value:

See the OKE documentation for the default value.

References:

1. https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks

Additional Information:

The lack of certificate revocation was flagged up as a high risk issue in the recent Kubernetes security audit. Without this feature, client certificate authentication is not suitable for end users.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1098	TA0003, TA0006	

2.2 Logging

2.2.1 Ensure access to OCI Audit service Log for OKE (Manual)

Profile Applicability:

- Level 1

Description:

The audit logs are part of the OKE managed Kubernetes control plane logs managed by OKE. OKE integrates with Oracle Cloud Infrastructure Audit Service.

All operations performed by the Kubernetes API server are visible as log events on the Oracle Cloud Infrastructure Audit service.

Rationale:

Logging is a crucial detective control for all systems to detect potential unauthorized access.

Impact:

The Control plane audit logs are managed by OKE. OKE Control plane logs are written to the Oracle Cloud Infrastructure Audit Service. The Oracle Cloud Infrastructure Audit service automatically records calls to all supported Oracle Cloud Infrastructure public application programming interface (API) endpoints as log events.

Audit:

Using Oracle Cloud Infrastructure Console

To monitor and manage operations performed by Container Engine for Kubernetes on a particular cluster:

1. In the Console, open the navigation menu. Under **Solutions and Platform**, go to **Developer Services** and click **Kubernetes Clusters**.
2. Choose a **Compartment** you have permission to work in.
3. On the **Cluster List** page, click the cluster's name for which you want to monitor and manage operations.
4. The **Cluster** page shows information about the cluster.
5. Display the **Work Requests** tab, showing the recent operations performed on the cluster.

To view operations performed by Container Engine for Kubernetes and the Kubernetes API server as log events in the Oracle Cloud Infrastructure Audit service:

1. In the Console, open the navigation menu. Under **Governance and Administration**, go to **Governance** and click **Audit**.
2. Choose a **Compartment** you have permission to work in.
3. Search and filter to show the operations you're interested in:

- To view operations performed by Container Engine for Kubernetes, enter `ClustersAPI` in the **Keywords** field and click **Search**.
- To view operations performed by the Kubernetes API server, enter `OKE API Server Admin Access` in the **Keywords** field and click **Search**.

Remediation:

No remediation is necessary for this control.

Default Value:

By default, Kubernetes API server logs and Container Engine for Kubernetes audit events are sent to the Oracle Cloud Infrastructure Audit service. By default, the Audit Log retention period is 90 days.

References:

1. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Tasks/contengmonitoringoke.htm
3. https://docs.cloud.oracle.com/en-us/iaas/Content/Audit/Tasks/viewinglogevents.htm#Viewing_Audit_Log_Events
4. https://docs.cloud.oracle.com/en-us/iaas/Content/Audit/Tasks/settingretentionperiod.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.	●	●	●
v7	6.2 Activate audit logging Ensure that local logging has been enabled on all systems and networking devices.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1543	TA0003, TA0004	M1026

3 Worker Nodes

This section consists of security recommendations for the components that run on OKE worker nodes.

3.1 Worker Node Configuration Files

This section covers recommendations for configuration files on the Oracle OKE worker nodes.

3.1.1 Ensure that the kubelet-config.json file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

If `kubelet` is running, and if it is using a file-based kubelet-config.json file, ensure that the proxy kubelet-config.json file has permissions of `644` or more restrictive.

Rationale:

The `kubelet` kubelet-config.json file controls various parameters of the `kubelet` service in the worker node. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

It is possible to run `kubelet` with the kubeconfig parameters configured as a Kubernetes ConfigMap instead of a file. In this case, there is no proxy kubelet-config.json file.

Impact:

Ensuring that the `kubelet-config` file permissions are set to 644 or more restrictive significantly strengthens the security posture of the Kubernetes environment by preventing unauthorized modifications. This restricts write access to the `kubelet-config` file, ensuring only administrators can alter crucial `kubelet` configurations, thereby reducing the risk of malicious alterations that could compromise the cluster's integrity.

However, this configuration may slightly impact usability, as it limits the ability for non-administrative users to make quick adjustments to the `kubelet` settings. Administrators will need to balance security needs with operational flexibility, potentially requiring adjustments to workflows for managing `kubelet` configurations.

Audit:

Method 1

SSH to the worker nodes

To check to see if the Kubelet Service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate kubelet-config.json file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--kubeconfig /etc/kubernetes/kubelet-config.json` which is the location of the kubelet-config.json file.

Run this command to obtain the kubelet-config.json file permissions:

```
stat -c %a /etc/kubernetes/kubelet-config.json
```

The output of the above command gives you the kubelet-config.json file's permissions.

Verify that if a file is specified and it exists, the permissions are `644` or more restrictive.

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file permissions on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the permission level of the file:

```
ls -l /host/etc/kubernetes/kubelet-config.json
```

Verify that if a file is specified and it exists, the permissions are 644 or more restrictive.

Remediation:

Run the below command (based on the file location on your system) on the each worker node. For example,

```
chmod 644 <kubelet-config.json file>
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kube-proxy/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1083, T1222	TA0005, TA0007	M1022

3.1.2 Ensure that the kubelet-config.json file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

If `kubelet` is running, ensure that the file ownership of its kubelet-config.json file is set to `root:root`.

Rationale:

The kubelet-config.json file for `kubelet` controls various parameters for the `kubelet` service in the worker node. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Impact:

None

Audit:

Method 1

SSH to the worker nodes

To check to see if the Kubelet Service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate kubelet-config.json file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--kubeconfig /etc/kubernetes/kubelet-config.json` which is the location of the kubelet-config.json file.

Run this command to obtain the kubelet-config.json file ownership:

```
stat -c %U:%G /etc/kubernetes/kubelet-config.json
```

The output of the above command gives you the kubelet-config.json file's ownership. Verify that the ownership is set to `root:root`.

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file ownership on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the ownership of the file:

```
ls -l /host/etc/kubernetes/kubelet-config.json
```

The output of the above command gives you the kubelet-config.json file's ownership. Verify that the ownership is set to `root:root`.

Remediation:

Run the below command (based on the file location on your system) on the each worker node. For example,

```
chown root:root <kubelet-config.json file>
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kube-proxy/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●
v6	5.1 Minimize And Sparingly Use Administrative Privileges Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.			

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1083, T1222	TA0005, TA0007	M1026

3.1.3 Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that if the kubelet refers to a configuration file with the `--config` argument, that file has permissions of 644 or more restrictive.

Rationale:

The kubelet reads various parameters, including security settings, from a config file specified by the `--config` argument. If this file is specified you should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None.

Audit:

Method 1

First, SSH to the relevant worker node:

To check to see if the Kubelet Service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /etc/kubernetes/kubelet.conf` which is the location of the Kubelet config file.

Run the following command:

```
stat -c %a /etc/kubernetes/kubelet.conf
```

The output of the above command is the Kubelet config file's permissions. Verify that the permissions are `644` or more restrictive.

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file permissions on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the permission level of the file:

```
ls -l /host/etc/kubernetes/kubelet.conf
```

Verify that if a file is specified and it exists, the permissions are 644 or more restrictive.

Remediation:

Run the following command (using the config file location identied in the Audit step)

```
chmod 644 etc/kubernetes/kubelet.conf
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/

2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1083, T1222	TA0005, TA0007	M1022

3.1.4 Ensure that the kubelet configuration file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that if the kubelet refers to a configuration file with the `--config` argument, that file is owned by root:root.

Rationale:

The kubelet reads various parameters, including security settings, from a config file specified by the `--config` argument. If this file is specified you should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None.

Audit:

Method 1

First, SSH to the relevant worker node:

To check to see if the Kubelet Service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /etc/kubernetes/kubelet.conf` which is the location of the Kubelet config file.

Run the following command:

```
stat -c %U:%G /etc/kubernetes/kubelet.conf
```

The output of the above command is the Kubelet config file's ownership. Verify that the ownership is set to `root:root`

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file ownership on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the ownership of the file:

```
ls -l /etc/kubernetes/kubelet.conf
```

The output of the above command gives you the azure.json file's ownership. Verify that the ownership is set to `root:root`.

Remediation:

Run the following command (using the config file location identied in the Audit step)

```
chown root:root /etc/kubernetes/kubelet.conf
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/

2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1083, T1222	TA0005, TA0007	M1026

3.2 Kubelet

This section contains recommendations for kubelet configuration.

Kubelet settings may be configured using arguments on the running kubelet executable, or they may be taken from a Kubelet config file. If both are specified, the executable argument takes precedence.

To find the Kubelet config file, run the following command:

```
ps -ef | grep kubelet | grep config
```

If the `--kubeconfig` argument is present, this gives the location of the Kubelet config file. This config file could be in JSON or YAML format depending on your distribution.

3.2.1 Ensure that the --anonymous-auth argument is set to false (Automated)

Profile Applicability:

- Level 1

Description:

Disable anonymous requests to the Kubelet server.

Rationale:

When enabled, requests that are not rejected by other configured authentication methods are treated as anonymous requests. These requests are then served by the Kubelet server. You should rely on authentication to authorize access and disallow anonymous requests.

Impact:

Anonymous requests will be rejected.

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `authentication: anonymous: enabled` set to `false`.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--anonymous-auth=false`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `authentication... "anonymous":{"enabled":false}` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
--anonymous-auth=false
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `"authentication.*anonymous":{"enabled":false}"` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/admin/kubelet-authentication-authorization/#kubelet-authentication

3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1190, T1210	TA0001, TA0008	M1025

3.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)

Profile Applicability:

- Level 1

Description:

Do not allow all requests. Enable explicit authorization.

Rationale:

Kubelets, by default, allow all authenticated requests (even anonymous ones) without needing explicit authorization checks from the apiserver. You should restrict this behavior and only allow explicitly authorized requests.

Impact:

Unauthorized requests will be denied.

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `--authentication-mode`.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--authentication-mode=Webhook`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `authentication... "webhook":{"enabled":true}` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
--authorization-mode=Webhook
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `"authentication.*webhook":{"enabled":true}"` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/admin/kubelet-authentication-authorization/#kubelet-authentication

3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1133	TA0001	M1026

3.2.3 Ensure that the --client-ca-file argument is set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Enable Kubelet authentication using certificates.

Rationale:

The connections from the apiserver to the kubelet are used for fetching logs for pods, attaching (through kubectl) to running pods, and using the kubelet's port-forwarding functionality. These connections terminate at the kubelet's HTTPS endpoint. By default, the apiserver does not verify the kubelet's serving certificate, which makes the connection subject to man-in-the-middle attacks, and unsafe to run over untrusted and/or public networks. Enabling Kubelet certificate authentication ensures that the apiserver could authenticate the Kubelet before submitting any requests.

Impact:

You require TLS to be configured on apiserver as well as kubelets.

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `--client-ca-file` set to the location of the client certificate authority file.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--client-ca-file` argument exists and is set to the location of the client certificate authority file.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `authentication..x509":("clientCAFile":"/etc/kubernetes/ca.crt` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
--client-ca-file=/etc/kubernetes/ca.crt \
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `"authentication.*x509":("clientCAFile":"/etc/kubernetes/pki/ca.crt"` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/
3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).		●	●
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1022

3.2.4 Ensure that the --read-only-port argument is set to 0 (Manual)

Profile Applicability:

- Level 1

Description:

Disable the read-only port.

Rationale:

The Kubelet process provides a read-only API in addition to the main Kubelet API. Unauthenticated access is provided to this read-only API which could possibly retrieve potentially sensitive information about the cluster.

Impact:

Removal of the read-only port will require that any service which made use of it will need to be re-configured to use the main Kubelet API.

Audit:

If using a Kubelet configuration file, check that there is an entry for `--read-only-port=0`.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet service config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--read-only-port` argument exists and is set to `0`.

If the `--read-only-port` argument is not present, check that there is a Kubelet config file specified by `--config`. Check that if there is a `readOnlyPort` entry in the file, it is set to `0`.

Remediation:

If modifying the Kubelet config file, edit the kubelet.service file `/etc/sytemd/system/kubelet.service` and set the below parameter

```
--read-only-port=0
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1133	TA0001, TA0003	M1030, M1035

3.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated)

Profile Applicability:

- Level 1

Description:

Do not disable timeouts on streaming connections.

Rationale:

Setting idle timeouts ensures that you are protected against Denial-of-Service attacks, inactive connections and running out of ephemeral ports.

Note: By default, `--streaming-connection-idle-timeout` is set to 4 hours which might be too high for your environment. Setting this as appropriate would additionally ensure that such streaming connections are timed out after serving legitimate use cases.

Impact:

Long-lived connections could be interrupted.

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `--streaming-connection-idle-timeout` is not set to `0`.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--streaming-connection-idle-timeout` argument is not set to `0`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `"streamingConnectionIdleTimeout":"4h0m0s"` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
--streaming-connection-idle-timeout
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `"streamingConnectionIdleTimeout":` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://github.com/kubernetes/kubernetes/pull/18552

3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	10.5 Enable Anti-Exploitation Features Enable anti-exploitation features on enterprise assets and software, where possible, such as Microsoft® Data Execution Prevention (DEP), Windows® Defender Exploit Guard (WDEG), or Apple® System Integrity Protection (SIP) and Gatekeeper™.		●	●
v7	8.3 Enable Operating System Anti-Exploitation Features/ Deploy Anti-Exploit Technologies Enable anti-exploitation features such as Data Execution Prevention (DEP) or Address Space Layout Randomization (ASLR) that are available in an operating system or deploy appropriate toolkits that can be configured to apply protection to a broader set of applications and executables.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1490	TA0040	M1028

3.2.6 Ensure that the --make-iptables-util-chains argument is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Allow Kubelet to manage iptables.

Rationale:

Kubelets can automatically manage the required changes to iptables based on how you choose your networking options for the pods. It is recommended to let kubelets manage the changes to iptables. This ensures that the iptables configuration remains in sync with pods networking configuration. Manually configuring iptables with dynamic pod network configuration changes might hamper the communication between pods/containers and to the outside world. You might have iptables rules too restrictive or too open.

Impact:

Kubelet would manage the iptables on the system and keep it in sync. If you are using any other iptables management solution, then there might be some conflicts.

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `--make-iptables-util-chains` set to `true`.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--make-iptables-util-chains=true`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `authentication... "makeIPTablesUtilChains":true` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
--make-iptables-util-chains:true
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `"makeIPTablesUtilChains": true` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	2.5 Allowlist Authorized Software Use technical controls, such as application allowlisting, to ensure that only authorized software can execute or be accessed. Reassess bi-annually, or more frequently.		●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1190	TA0001	

3.2.7 Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Automated)

Profile Applicability:

- Level 1

Description:

Security relevant information should be captured. The `--event-qps` flag on the Kubelet can be used to limit the rate at which events are gathered. Setting this too low could result in relevant events not being logged, however the unlimited setting of `0` could result in a denial of service on the kubelet.

Rationale:

It is important to capture all events and not restrict event creation. Events are an important source of security information and analytics that ensure that your environment is consistently monitored using the event data.

Impact:

Setting this parameter to `0` could result in a denial of service condition due to excessive events being created. The cluster's event processing and storage systems should be scaled to handle expected event loads.

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `--event-qps` set to `0` or a value equal to or greater than 0.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--event-qps=0`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `"eventRecordQPS": 0` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
--event-qps=0
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `"eventRecordQPS"` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/apis/kubeletconfig/v1beta1/types.go
3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.	●	●	●
v7	6.2 Activate audit logging Ensure that local logging has been enabled on all systems and networking devices.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1543	TA0003, TA0004	M1047

3.2.8 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Setup TLS connection on the Kubelets.

Rationale:

Kubelet communication contains sensitive parameters that should remain encrypted in transit. Configure the Kubelets to serve only HTTPS traffic.

Impact:

TLS and client certificate authentication must be configured for your Kubernetes cluster deployment.

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `tls-cert-file` set to `correct pem file` and `tls-private-key-file` is set to `correct key file`

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `tls-cert-file=/var/lib/kubelet/pki/tls.pem`. Verify that the `tls-private-key-file=/var/lib/kubelet/pki/tls.key`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `tlsCertFile` and `tlsPrivateKeyFile` are set by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
Verify that the `tls-cert-file=/var/lib/kubelet/pki/tls.pem`.
Verify that the `tls-private-key-file=/var/lib/kubelet/pki/tls.key`.
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `tlsCertFile` and `tlsPrivateKeyFile` are set by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. http://rootsquash.com/2016/05/10/securing-the-kubernetes-api/

3. https://github.com/kelseyhightower/docker-kubernetes-tls-guide
4. https://jvns.ca/blog/2017/08/05/how-kubernetes-certificates-work/
5. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).		●	●
v7	14.3 Disable Workstation to Workstation Communication Disable all workstation to workstation communication to limit an attacker's ability to move laterally and compromise neighboring systems, through technologies such as Private VLANs or microsegmentation.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1552	TA0001, TA0006	M1035, M1041

3.2.9 Ensure that the --rotate-certificates argument is not set to false (Automated)

Profile Applicability:

- Level 1

Description:

Enable kubelet client certificate rotation.

Rationale:

The `--rotate-certificates` setting causes the kubelet to rotate its client certificates by creating new CSRs as its existing credentials expire. This automated periodic rotation ensures that the there is no downtime due to expired certificates and thus addressing availability in the CIA security triad.

Note: This recommendation only applies if you let kubelets get their certificates from the API server. In case your kubelet certificates come from an outside authority/tool (e.g. Vault) then you need to take care of rotation yourself.

Note: This feature also require the `RotateKubeletClientCertificate` feature gate to be enabled (which is the default since Kubernetes v1.7)

Impact:

None

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `--rotate-certificates` set to `true`.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--rotate-certificates` is present.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `rotateCertificates` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
Verify that the `--rotate-certificates` is present.
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `rotateCertificates` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://github.com/kubernetes/kubernetes/pull/41912
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-tls-bootstrapping/#kubelet-configuration
3. https://kubernetes.io/docs/imported/release/notes/
4. https://kubernetes.io/docs/reference/command-line-tools-reference/feature-gates/
5. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).		●	●
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1022

3.2.10 Ensure that the --rotate-server-certificates argument is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Enable kubelet server certificate rotation.

Rationale:

`--rotate-server-certificates` causes the kubelet to both request a serving certificate after bootstrapping its client credentials and rotate the certificate as its existing credentials expire. This automated periodic rotation ensures that the there are no downtimes due to expired certificates and thus addressing availability in the CIA security triad.

Note: This recommendation only applies if you let kubelets get their certificates from the API server. In case your kubelet certificates come from an outside authority/tool (e.g. Vault) then you need to take care of rotation yourself.

Impact:

None

Audit:

Audit Method 1:

If using a Kubelet configuration file, check that there is an entry for `--rotate-server-certificates` is set to `true`.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

```
find / -name kubelet.service
```

The output of the above command should return the file and location `/etc/systemd/system/kublet.service` which is the location of the Kubelet service config file.

Open the Kubelet service config file:

```
sudo more etc/systemd/system/kublet.service
```

Verify that the `--rotate-server-certificates=true`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `--rotate-server-certificates` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number" NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet service config file, edit the kubelet.service file `/etc/systemd/system/kubelet.service` and set the below parameter

```
--rotate-server-certificates=true
```

Remediation Method 2:

If using the api configz endpoint consider searching for the status of `--rotate-server-certificates` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=10.0.10.4 (example node name from "kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all remediations: Based on your system, restart the `kubelet` service and check status

```
systemctl daemon-reload
systemctl restart kubelet.service
systemctl status kubelet -l
```

Default Value:

See the OKE documentation for the default value.

References:

1. https://github.com/kubernetes/kubernetes/pull/45059

2. https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/#kubelet-configuration
3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm
4. https://kubernetes.io/docs/reference/access-authn-authz/kubelet-tls-bootstrapping/#certificate-rotation

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).		●	●
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1022

4 Policies

This section contains recommendations for various Kubernetes policies which are important to the security of Oracle OKE customer environment.

4.1 RBAC and Service Accounts

4.1.1 Ensure that the cluster-admin role is only used where required (Manual)

Profile Applicability:

- Level 1

Description:

The RBAC role `cluster-admin` provides wide-ranging powers over the environment and should be used only where and when needed.

Rationale:

Kubernetes provides a set of default roles where RBAC is used. Some of these roles such as `cluster-admin` provide wide-ranging privileges which should only be applied where absolutely necessary. Roles such as `cluster-admin` allow super-user access to perform any action on any resource. When used in a `ClusterRoleBinding`, it gives full control over every resource in the cluster and in all namespaces. When used in a `RoleBinding`, it gives full control over every resource in the rolebinding's namespace, including the namespace itself.

Impact:

Care should be taken before removing any `clusterrolebindings` from the environment to ensure they were not required for operation of the cluster. Specifically, modifications should not be made to `clusterrolebindings` with the `system:` prefix as they are required for the operation of system components.

Audit:

Obtain a list of the principals who have access to the `cluster-admin` role by reviewing the `clusterrolebinding` output for each role binding that has access to the `cluster-admin` role.

Here's a concise and effective CLI statement to list all principals (users, groups, or service accounts) bound to the cluster-admin role:

```
kubectl get clusterrolebinding -o jsonpath='{range
.items[?(@.roleRef.name=="cluster-admin")]}{.metadata.name}{"\n"}{range
.subjects[*]}{.kind}{"\t"}{.name}{"\n"}{end}{"\n"}{end}'
```

Review each principal listed and ensure that `cluster-admin` privilege is required for it.

Remediation:

Identify all clusterrolebindings to the cluster-admin role. Check if they are used and if they need this role or if they could use a role with fewer privileges.

Where possible, first bind users to a lower privileged role and then remove the clusterrolebinding to the cluster-admin role :

```
kubectl delete clusterrolebinding [name]
```

Default Value:

By default a single `clusterrolebinding` called `cluster-admin` is provided with the `system:masters` group as its principal.

References:

1. https://kubernetes.io/docs/admin/authorization/rbac/#user-facing-roles
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.002	TA0001, TA0004	M1026

4.1.2 Minimize access to secrets (Manual)

Profile Applicability:

- Level 1

Description:

The Kubernetes API stores secrets, which may be service account tokens for the Kubernetes API or credentials used by workloads in the cluster. Access to these secrets should be restricted to the smallest possible group of users to reduce the risk of privilege escalation.

Rationale:

Inappropriate access to secrets stored within the Kubernetes cluster can allow for an attacker to gain additional access to the Kubernetes cluster or external resources whose credentials are stored as secrets.

Impact:

Care should be taken not to remove access to secrets to system components which require this for their operation

Audit:

Review the users who have `get`, `list` or `watch` access to `secrets` objects in the Kubernetes API.

Below is a command to print which objects have get, list or watch granted for each matching role including roles that grant access via wildcards like resources: ["*"*,"secrets/*"] or verbs: ["*"]:

```
kubectl get clusterrole,role -A -o json | jq -r '
  def wanted: ["get","list","watch"];
  .items[] as $r
  | [ $r.rules[]?
      | select(
          ((.apiGroups? // [""]) | any(.=="" or .=="*"))
          and ((.resources? // []) | any(.=="secrets" or .=="secrets/*" or
.=="*"))
          and ((.verbs? // []) | any(.=="*" or .=="get" or .=="list" or
.=="watch"))
        )
      | if ((.verbs? // []) | any(.=="*"))
        then wanted[] else (.verbs[]? | select(IN("get","list","watch"))) end
    ] as $verbs
  | select($verbs | length > 0)
  | "\($r.kind): \($r.metadata.name) (namespace: \($r.metadata.namespace //
"cluster-wide")) | verbs: \($verbs | unique | join(","))"
'
```

Sample output:

```
ClusterRole: admin (namespace: cluster-wide) | verbs: get,list,watch
ClusterRole: cluster-admin (namespace: cluster-wide) | verbs: get,list,watch
ClusterRole: edit (namespace: cluster-wide) | verbs: get,list,watch
ClusterRole: system:aggregate-to-edit (namespace: cluster-wide) | verbs:
get,list,watch
ClusterRole: system:cloud-controller-manager (namespace: cluster-wide) |
verbs: get,list,watch
ClusterRole: system:controller:generic-garbage-collector (namespace: cluster-
wide) | verbs: get,list,watch
ClusterRole: system:controller:namespace-controller (namespace: cluster-wide)
| verbs: get,list,watch
ClusterRole: system:controller:resourcequota-controller (namespace: cluster-
wide) | verbs: list,watch
ClusterRole: system:gcp-controller-manager (namespace: cluster-wide) | verbs:
get,list,watch
ClusterRole: system:gke-common-webhooks (namespace: cluster-wide) | verbs:
get,list,watch
ClusterRole: system:glbc-status (namespace: cluster-wide) | verbs: get
ClusterRole: system:kube-controller-manager (namespace: cluster-wide) |
verbs: get,list,watch
ClusterRole: system:kubestore-collector (namespace: cluster-wide) | verbs:
get,list,watch
ClusterRole: system:node (namespace: cluster-wide) | verbs: get,list,watch
Role: operator (namespace: gmp-public) | verbs: get,list,watch
Role: operator (namespace: gmp-system) | verbs: get,list,watch
Role: system:controller:bootstrap-signer (namespace: kube-system) | verbs:
get,list,watch
Role: system:controller:token-cleaner (namespace: kube-system) | verbs:
get,list,watch
```

Remediation:

Where possible, remove `get`, `list` or `watch` access to `secret` objects in the cluster.

Default Value:

By default, the following list of principals have `get` privileges on `secret` objects

```
CLUSTERROLEBINDING                                          SUBJECT
TYPE            SA-NAMESPACE
cluster-admin                                               system:masters
Group
system:controller:clusterrole-aggregation-controller  clusterrole-
aggregation-controller  ServiceAccount  kube-system
system:controller:expand-controller                         expand-controller
ServiceAccount  kube-system
system:controller:generic-garbage-collector                 generic-garbage-
collector           ServiceAccount  kube-system
system:controller:namespace-controller                      namespace-controller
ServiceAccount  kube-system
system:controller:persistent-volume-binder                  persistent-volume-
binder            ServiceAccount  kube-system
system:kube-controller-manager                              system:kube-controller-
manager      User
```

References:

1. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1026

4.1.3 Minimize wildcard use in Roles and ClusterRoles (Manual)

Profile Applicability:

- Level 1

Description:

Kubernetes Roles and ClusterRoles provide access to resources based on sets of objects and actions that can be taken on those objects. It is possible to set either of these to be the wildcard "*" which matches all items.

Use of wildcards is not optimal from a security perspective as it may allow for inadvertent access to be granted when new resources are added to the Kubernetes API either as CRDs or in later versions of the product.

Rationale:

The principle of least privilege recommends that users are provided only the access required for their role and nothing more. The use of wildcard rights grants is likely to provide excessive rights to the Kubernetes API.

Audit:

Retrieve the roles defined across each namespaces in the cluster and review for wildcards

Here’s a null-safe, column-formatted command that shows only Roles and ClusterRoles that use a wildcard (*) anywhere in verbs, resources, or apiGroups—and tells you which field(s) use the wildcard:

```
kubectl get clusterrole,role -A -o json | jq -r '
  def has_star(a): (a // []) | any(. == "*");

  .items[]
  | . as $r
  | ( any($r.rules[]?; has_star(.verbs)) )      as $wv
  | ( any($r.rules[]?; has_star(.resources)) )  as $wr
  | ( any($r.rules[]?; has_star(.apiGroups)) )  as $wg
  | select($wv or $wr or $wg)
  | [
      $r.kind,
      $r.metadata.name,
      ($r.metadata.namespace // "cluster-wide"),
      ([ if $wv then "verbs" else empty end,
         if $wr then "resources" else empty end,
         if $wg then "apiGroups" else empty end ] | join(","))
    ]
  | @tsv
' | awk -F'\t' 'BEGIN {
  OFS="\t";
  printf "%-15s %-40s %-20s %-20s\n", "KIND", "NAME", "NAMESPACE",
"WILDCARD_IN"
  print  "--------------- ---------------------------------------- ----------
---------- --------------------"
} {
  printf "%-15s %-40s %-20s %-20s\n", $1, $2, $3, $4
}'
```

Sample Output from command:

```
KIND            NAME                                         NAMESPACE
WILDCARD_IN
--------------- ----------------------------------------     -----------------
--- --------------------
ClusterRole     cluster-admin                                cluster-wide
verbs,resources,apiGroups
ClusterRole     system:controller:disruption-controller      cluster-wide
apiGroups
ClusterRole     system:controller:generic-garbage-collector cluster-wide
resources,apiGroups
ClusterRole     system:controller:horizontal-pod-autoscaler cluster-wide
resources,apiGroups
ClusterRole     system:controller:namespace-controller       cluster-wide
resources,apiGroups
ClusterRole     system:controller:resourcequota-controller   cluster-wide
resources,apiGroups
ClusterRole     system:kube-controller-manager               cluster-wide
resources,apiGroups
ClusterRole     system:kubelet-api-admin                     cluster-wide
verbs
```

Remediation:

Where possible replace any use of wildcards in clusterroles and roles with specific objects or actions.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.002	TA0001, TA0004	M1026

4.1.4 Minimize access to create pods (Manual)

Profile Applicability:

- Level 1

Description:

The ability to create pods in a namespace can provide a number of opportunities for privilege escalation, such as assigning privileged service accounts to these pods or mounting hostPaths with access to sensitive data (unless Pod Security Policies are implemented to restrict this access)

As such, access to create new pods should be restricted to the smallest possible group of users.

Rationale:

The ability to create pods in a cluster opens up possibilities for privilege escalation and should be restricted, where possible.

Impact:

Care should be taken not to remove access to pods to system components which require this for their operation

Audit:

OKE is a standard Kubernetes cluster under the hood, so we can use kubectl auth can-i together with rolebinding lookups to get this info.

Run the following command to list all the users and service accounts that have access to create pod objects in the Kubernetes API:

```
echo "=== Users ==="
for user in $(kubectl get clusterrolebinding -o jsonpath='{range
.items[*].subjects[?(@.kind=="User")]}{.name}{"\n"}{end}'); do
  if kubectl auth can-i create pods --as="$user" >/dev/null 2>&1; then
    echo "$user"
  fi
done

echo -e "\n=== Service Accounts ==="
for sa in $(kubectl get clusterrolebinding,rolebinding -A -o jsonpath='{range
.items[*].subjects[?(@.kind=="ServiceAccount")]}{.namespace}:{.name}{"\n"}{en
d}'); do
  ns=$(echo "$sa" | cut -d: -f1)
  name=$(echo "$sa" | cut -d: -f2)
  if kubectl auth can-i create pods --as=system:serviceaccount:$ns:$name
>/dev/null 2>&1; then
    echo "$sa"
  fi
done
```

Sample output:

```
=== Users ===
cluster-admin

=== Service Accounts ===
kube-system:daemon-set-controller
kube-system:job-controller
kube-system:persistent-volume-binder
kube-system:replicaset-controller
kube-system:replication-controller
kube-system:statefulset-controller
```

- kubectl get clusterrolebinding and rolebinding return which users or service accounts are bound to roles.
- kubectl auth can-i create pods --as= checks effective permissions.
- The script loops over those bindings and prints only those who truly can create pods.

Remediation:

Where possible, remove `create` access to `pod` objects in the cluster.

Default Value:

By default, the following list of principals have `create` privileges on `pod` objects

```
CLUSTERROLEBINDING                              SUBJECT                  TYPE
SA-NAMESPACE
cluster-admin                                   system:masters           Group
-
system:controller:daemon-set-controller                 daemon-set-controller
        ServiceAccount  kube-system
system:controller:job-controller                job-controller
ServiceAccount  kube-system
system:controller:persistent-volume-binder      persistent-volume-binder
        ServiceAccount  kube-system
system:controller:replicaset-controller                 replicaset-controller
        ServiceAccount  kube-system
system:controller:replication-controller        replication-controller
        ServiceAccount  kube-system
system:controller:statefulset-controller        statefulset-controller
        ServiceAccount  kube-system
```

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.002	TA0001, TA0004	M1026

4.1.5 Ensure that default service accounts are not actively used. (Automated)

Profile Applicability:

- Level 1

Description:

The `default` service account should not be used to ensure that rights granted to applications can be more easily audited and reviewed.

Rationale:

Kubernetes provides a `default` service account which is used by cluster workloads where no specific service account is assigned to the pod.

Where access to the Kubernetes API from a pod is required, a specific service account should be created for that pod, and rights granted to that service account.

The default service account should be configured such that it does not provide a service account token and does not have any explicit rights assignments.

Impact:

All workloads which require access to the Kubernetes API will require an explicit service account to be created.

Audit:

For each namespace in the cluster, review the rights assigned to the default service account and ensure that it has no roles or cluster roles bound to it apart from the defaults.

Additionally ensure that the `automountServiceAccountToken: false` setting is in place for each default service account.

```
kubectl get serviceaccounts/default -o yaml
```

Sample Output:

```
apiVersion: v1
automountServiceAccountToken: false
kind: ServiceAccount
metadata:
  creationTimestamp: "2025-10-18T16:11:09Z"
  name: default
  namespace: default
...
```

Remediation:

Create explicit service accounts wherever a Kubernetes workload requires specific access to the Kubernetes API server.

Modify the configuration of each default service account to include this value:
`automountServiceAccountToken: false`

Automatic remediation for the default account:

```
kubectl patch serviceaccount default -p $'automountServiceAccountToken:
false'
```

Default Value:

By default the `default` service account allows for its service account token to be mounted in pods in its namespace.

References:

1. https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.3 Disable Dormant Accounts Delete or disable any dormant accounts after a period of 45 days of inactivity, where supported.	●	●	●
v7	16.9 Disable Dormant Accounts Automatically disable dormant accounts after a set period of inactivity.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1028

4.1.6 Ensure that Service Account Tokens are only mounted where necessary (Automated)

Profile Applicability:

- Level 1

Description:

Service accounts tokens should not be mounted in pods except where the workload running in the pod explicitly needs to communicate with the API server

Rationale:

Mounting service account tokens inside pods can provide an avenue for privilege escalation attacks where an attacker is able to compromise a single pod in the cluster.

Avoiding mounting these tokens removes this attack avenue.

Impact:

Pods mounted without service account tokens will not be able to communicate with the API server, except where the resource is available to unauthenticated principals.

Audit:

Review pod and service account objects in the cluster and ensure that the option below is set, unless the resource explicitly requires this access.

```
automountServiceAccountToken: false
echo "=== ServiceAccounts with automountServiceAccountToken=false ==="
printf "%-25s %-40s\n" "NAMESPACE" "SERVICEACCOUNT"
echo "------------------------- ----------------------------------------"
kubectl get sa -A -o jsonpath='{range
.items[?(@.automountServiceAccountToken==false)]}{.metadata.namespace}{"\t"}{
.metadata.name}{"\n"}{end}' \
| sort | awk -F'\t' '{printf "%-25s %-40s\n", $1, $2}'

echo ""
echo "=== Pods with automountServiceAccountToken=false ==="
printf "%-25s %-40s\n" "NAMESPACE" "POD"
echo "------------------------- ----------------------------------------"
kubectl get pods -A -o jsonpath='{range
.items[?(@.spec.automountServiceAccountToken==false)]}{.metadata.namespace}{"
\t"}{.metadata.name}{"\n"}{end}' \
| sort | awk -F'\t' '{printf "%-25s %-40s\n", $1, $2}'
```

Note:

- kubectl get ... -A scans all namespaces.
- jsonpath filters for `automountServiceAccountToken == false`.
- awk formats the output into aligned columns.

- If a section is empty, no objects explicitly set `automountServiceAccountToken: false` — meaning they still auto-mount tokens by default, and may need remediation.

Sample Output:

```
=== ServiceAccounts with automountServiceAccountToken=false ===
NAMESPACE                 SERVICEACCOUNT
------------------------- ---------------------------------------
default                   default
kube-system               metrics-reader

=== Pods with automountServiceAccountToken=false ===
NAMESPACE                 POD
------------------------- ---------------------------------------
dev                       api-service
prod                      nginx-test
```

Remediation:

Modify the definition of pods and service accounts which do not need to mount service account tokens to disable it.

Default Value:

By default, all pods get a service account token mounted in them.

References:

1. https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1528, T1555	TA0006	M1026

4.2 Pod Security Policies

Pod Security Standards (PSS) are recommendations for securing deployed workloads to reduce the risks of container breakout. There are a number of ways if implementing PSS, including the built-in Pod Security Admission controller, or external policy control systems which integrate with Kubernetes via validating and mutating webhooks.

The previous feature described in this document, pod security policy (preview), was deprecated with version 1.21, and removed as of version 1.25. After pod security policy (preview) is deprecated, you must disable the feature on any existing clusters using the deprecated feature to perform future cluster upgrades and stay within Oracle support.

4.2.1 Minimize the admission of privileged containers (Automated)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `securityContext.privileged` flag set to `true`.

Rationale:

Privileged containers have access to all Linux Kernel capabilities and devices. A container running with full privileges can do almost everything that the host can do. This flag exists to allow special use-cases, like manipulating the network stack and accessing devices.

There should be at least one admission control policy defined which does not permit privileged containers.

If you need to run privileged containers, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.containers[].securityContext.privileged: true`, `spec.initContainers[].securityContext.privileged: true` and `spec.ephemeralContainers[].securityContext.privileged: true` will not be permitted.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of privileged containers.

Since manually searching through each pod's configuration might be tedious, especially in environments with many pods, you can use a more automated approach with grep or other command-line tools.

Here's an example of how you might approach this with a combination of kubectl, grep, and shell scripting for a more automated solution:

```
kubectl get pods -A -o json \
| jq -r '
  .items[]
  | .metadata.namespace as $ns
  | .metadata.name as $pod
  | .spec.containers[]? as $c
  | select(($c.securityContext.privileged // false) == true)
  | "\($ns)/\($pod)\tcontainer=\($c.name)\tprivileged=true"
'
```

Sample Output:

```
kube-system/csi-oci-node-hsr28         container=csi-node-driver
       privileged=true
kube-system/kube-proxy-g949b           container=kube-proxy
privileged=true
kube-system/proxymux-client-5p85m      container=proxymux-client
       privileged=true
kube-system/vcn-native-ip-cni-jwxtl    container=install-cni-ips
       privileged=true
kube-system/vcn-native-ip-cni-jwxtl    container=oci-cni-device-plugin
       privileged=true
```

OR

```
kubectl get pods --all-namespaces -o json | jq '.items[] |
select(.metadata.namespace != "kube-system" and
.spec.containers[]?.securityContext?.privileged == true) | {pod:
.metadata.name, namespace: .metadata.namespace, container:
.spec.containers[].name}'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command uses jq, a command-line JSON processor, to parse the JSON output from kubectl get pods and filter out pods where any container has the securityContext.privileged flag set to true. Please note that you might need to adjust the command depending on your specific requirements and the structure of your pod specifications.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of privileged containers.

To enable PSA for a namespace in your cluster, set the pod-security.kubernetes.io/enforce label with the policy value you want to enforce.

```
kubectl label --overwrite ns NAMESPACE pod-
security.kubernetes.io/enforce=restricted
```

The above command enforces the restricted policy for the NAMESPACE namespace.

You can also enable Pod Security Admission for all your namespaces. For example:

```
kubectl label --overwrite ns --all pod-
security.kubernetes.io/warn=baseline
```

Default Value:

By default, there are no restrictions on the creation of privileged containers.

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-admission/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1611	TA0004	M1038, M1048

4.2.2 Minimize the admission of containers wishing to share the host process ID namespace (Automated)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `hostPID` flag set to true.

Rationale:

A container running in the host's PID namespace can inspect processes running outside the container. If the container also has access to ptrace capabilities this can be used to escalate privileges outside of the container.

There should be at least one admission control policy defined which does not permit containers to share the host PID namespace.

If you need to run containers which require hostPID, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.hostPID: true` will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostPID` containers

Search for the hostPID Flag: In the YAML output, look for the `hostPID` setting under the spec section to check if it is set to `true`.

```
kubectl get pods -A -o json \
| jq -r '
  .items[]
  | select((.spec.hostPID // false) == true)
  | "\(.metadata.namespace)/\(.metadata.name)"
'
```

OR check for all values set:

```
kubectl get pods -A -o json \
| jq -r '.items[] |
"\(.metadata.namespace)/\(.metadata.name)\t\(.spec.hostPID // false)"'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command retrieves all pods across all namespaces in JSON format, then uses jq to filter out those with the `hostPID` flag set to `true`, and finally formats the output to show the namespace and name of each matching pod.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostPID` containers.

Default Value:

By default, there are no restrictions on the creation of `hostPID` containers.

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-admission/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13.10 Perform Application Layer Filtering Perform application layer filtering. Example implementations include a filtering proxy, application layer firewall, or gateway.			●
v7	12.9 Deploy Application Layer Filtering Proxy Server Ensure that all network traffic to or from the Internet passes through an authenticated application layer proxy that is configured to filter unauthorized connections.			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.002	TA0001, TA0004	M1026

4.2.3 Minimize the admission of containers wishing to share the host IPC namespace (Automated)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `hostIPC` flag set to true.

Rationale:

A container running in the host's IPC namespace can use IPC to interact with processes outside the container.

There should be at least one admission control policy defined which does not permit containers to share the host IPC namespace.

If you need to run containers which require hostIPC, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.hostIPC: true` will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostIPC` containers

Search for the hostIPC Flag: In the YAML output, look for the `hostIPC` setting under the spec section to check if it is set to `true`.

```
kubectl get pods -A -o json \
| jq -r '
  .items[]
  | select((.spec.hostIPC // false) == true)
  | "\(.metadata.namespace)/\(.metadata.name)\thostIPC=true"
'
```

OR to include pod UID and node name:

```
kubectl get pods -A -o json \
| jq -r '
  .items[]
  | select((.spec.hostIPC // false) == true)
  | "\(.metadata.namespace)/\(.metadata.name)\tnode=\(.spec.nodeName // "-
")\thostIPC=true"
'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command retrieves all pods across all namespaces in JSON format, then uses jq to filter out those with the `hostIPC` flag set to `true`, and finally formats the output to show the namespace and name of each matching pod.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostIPC` containers.

Default Value:

By default, there are no restrictions on the creation of `hostIPC` containers.

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-admission/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.12 Segment Data Processing and Storage Based on Sensitivity Segment data processing and storage based on the sensitivity of the data. Do not process sensitive data on enterprise assets intended for lower sensitivity data.		●	●
v7	14.1 Segment the Network Based on Sensitivity Segment the network based on the label or classification level of the information stored on the servers, locate all sensitive information on separated Virtual Local Area Networks (VLANs).		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1098	TA0003	M1030

4.2.4 Minimize the admission of containers wishing to share the host network namespace (Automated)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `hostNetwork` flag set to true.

Rationale:

A container running in the host's network namespace could access the local loopback device, and could access network traffic to and from other pods.

There should be at least one admission control policy defined which does not permit containers to share the host network namespace.

If you need to run containers which require access to the host's network namespaces, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.hostNetwork: true` will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostNetwork` containers

Given that manually checking each pod can be time-consuming, especially in large environments, you can use a more automated approach to filter out pods where `hostNetwork` is set to `true`. Here's a command using kubectl and jq:

```
kubectl get pods -A -o json \
| jq -r '
  .items[]
  | select((.spec.hostNetwork // false) == true)
  | "\(.metadata.namespace)/\(.metadata.name)\thostNetwork=true"
'
```

OR to include node info:

```
kubectl get pods -A -o json \
| jq -r '
  .items[]
  | select((.spec.hostNetwork // false) == true)
  | "\(.metadata.namespace)/\(.metadata.name)\tnode=\(.spec.nodeName // "-
")\thostNetwork=true"
'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command retrieves all pods across all namespaces in JSON format, then uses jq to filter out those with the `hostNetwork` flag set to `true`, and finally formats the output to show the namespace and name of each matching pod.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostNetwork` containers.

Default Value:

By default, there are no restrictions on the creation of `hostNetwork` containers.

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-admission/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.12 Segment Data Processing and Storage Based on Sensitivity Segment data processing and storage based on the sensitivity of the data. Do not process sensitive data on enterprise assets intended for lower sensitivity data.		●	●
v7	14.1 Segment the Network Based on Sensitivity Segment the network based on the label or classification level of the information stored on the servers, locate all sensitive information on separated Virtual Local Area Networks (VLANs).		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1098	TA0003	M1030

4.2.5 Minimize the admission of containers with allowPrivilegeEscalation (Automated)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `allowPrivilegeEscalation` flag set to `true`. Allowing this right can lead to a process running a container getting more rights than it started with.

It's important to note that these rights are still constrained by the overall container sandbox, and this setting does not relate to the use of privileged containers.

Rationale:

A container running with the `allowPrivilegeEscalation` flag set to `true` may have processes that can gain more privileges than their parent.

There should be at least one admission control policy defined which does not permit containers to allow privilege escalation. The option exists (and is defaulted to true) to permit setuid binaries to run.

If you have need to run containers which use setuid binaries or require privilege escalation, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.allowPrivilegeEscalation: true` will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of containers which allow privilege escalation.

This command gets all pods across all namespaces, outputs their details in JSON format, and uses jq to parse and filter the output for containers with `allowPrivilegeEscalation` set to `true`. Here is a command to list all pods (across all namespaces) where any container has `securityContext.allowPrivilegeEscalation: true`:

```
kubectl get pods -A -o json \
| jq -r '
  .items[]
  | .metadata.namespace as $ns
  | .metadata.name as $pod
  | (.spec.initContainers[]?, .spec.containers[]?) as $c
  | ($c.securityContext.allowPrivilegeEscalation // false) as $ape
  | select($ape == true)
  | "\($ns)/\($pod)\tcontainer=\($c.name)\tallowPrivilegeEscalation=true"
'
```

- -A — checks all namespaces.
- (.spec.initContainers[]?, .spec.containers[]?) — inspects both regular and init containers.
- .securityContext.allowPrivilegeEscalation — the field that controls privilege escalation.
- // false — safely defaults to false if the field is missing.

OR to filter out system namespaces in OKE:

```
EXCLUDE_NS='["kube-system","kube-public","oci-system","oci-service-operator-
system","gatekeeper-system"]'

kubectl get pods -A -o json \
| jq -r --argjson ex "$EXCLUDE_NS" '
  .items[]
  | select( (.metadata.namespace as $ns | $ex | index($ns) | not) )
  | .metadata.namespace as $ns
  | .metadata.name as $pod
  | (.spec.initContainers[]?, .spec.containers[]?) as $c
  | ($c.securityContext.allowPrivilegeEscalation // false) as $ape
  | select($ape == true)
  | "\($ns)/\($pod)\tcontainer=\($c.name)\tallowPrivilegeEscalation=true"
'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command uses jq, a command-line JSON processor, to parse the JSON output from kubectl get pods and filter out pods where any container has the securityContext.privileged flag set to true. Please note that you might need to adjust the command depending on your specific requirements and the structure of your pod specifications.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of containers with `.spec.allowPrivilegeEscalation` set to `true`.

Default Value:

By default, there are no restrictions on contained process ability to escalate privileges, within the context of the container.

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-admission/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1611	TA0004	M1038, M1048

4.3 CNI Plugin

4.3.1 Ensure latest CNI version is used (Automated)

Profile Applicability:

- Level 1

Description:

There are a variety of CNI plugins available for Kubernetes. If the CNI in use does not support Network Policies it may not be possible to effectively restrict traffic in the cluster.

Rationale:

Kubernetes network policies are enforced by the CNI plugin in use. As such it is important to ensure that the CNI plugin supports both Ingress and Egress network policies.

Impact:

None.

Audit:

Review the documentation of CNI plugin in use by the cluster, and confirm that it supports network policies.

Check the DaemonSets in the kube-system namespace: Many CNI plugins operate as DaemonSets within the kube-system namespace. To see what's running:

```
kubectl get daemonsets -n kube-system
```

Look for known CNI providers like Calico, Flannel, Cilium, etc.

You can further inspect the configuration of these DaemonSets to understand more about the CNI setup:

```
kubectl get daemonsets -n kube-system -o json \
| jq -r '
  .items[]
  | select(
      (.metadata.name | test("cni|calico|flannel|weave|cilium|canal|multus";
"i"))
      or
      ([.spec.template.spec.containers[].image] | join(" ") |
test("cni|calico|flannel|weave|cilium|canal|multus"; "i"))
    )
  | "\(.metadata.name)\timage=\(.spec.template.spec.containers[].image)"
'
```

Check the CNI Configuration Files: If you have access to the nodes (via SSH), you can check the CNI configuration directly in /etc/cni/net.d/. This often requires node-level access, which might not be available depending on your permissions and the security setup of your environment.

Remediation:

As with RBAC policies, network policies should adhere to the policy of least privileged access. Start by creating a deny all policy that restricts all inbound and outbound traffic from a namespace or create a global policy using Calico.

Default Value:

This will depend on the CNI plugin in use.

References:

1. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/
2. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

Additional Information:

One example here is Flannel (https://github.com/coreos/flannel) which does not support Network policy unless Calico is also in use.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.4 Perform Automated Application Patch Management Perform application updates on enterprise assets through automated patch management on a monthly, or more frequent, basis.	●	●	●
v7	11.4 Install the Latest Stable Version of Any Security-related Updates on All Network Devices Install the latest stable version of any security-related updates on all network devices.	●	●	●

4.3.2 Ensure that all Namespaces have Network Policies defined (Manual)

Profile Applicability:

- Level 1

Description:

Use network policies to isolate traffic in your cluster network.

Rationale:

Running different applications on the same Kubernetes cluster creates a risk of one compromised application attacking a neighboring application. Network segmentation is important to ensure that containers can communicate only with those they are supposed to. A network policy is a specification of how selections of pods are allowed to communicate with each other and other network endpoints.

Network Policies are namespace scoped. When a network policy is introduced to a given namespace, all traffic not allowed by the policy is denied. However, if there are no network policies in a namespace all traffic will be allowed into and out of the pods in that namespace.

Impact:

Once network policies are in use within a given namespace, traffic not explicitly allowed by a network policy will be denied. As such it is important to ensure that, when introducing network policies, legitimate traffic is not blocked.

Audit:

Run the below command and review the `NetworkPolicy` objects created in the cluster. This command will list all namespaces in your OKE cluster that have one or more NetworkPolicy objects defined:

```
echo -e "NAMESPACE\tNETWORKPOLICY_COUNT" && \
for ns in $(kubectl get ns -o jsonpath='{.items[*].metadata.name}'); do
  count=$(kubectl get networkpolicy -n "$ns" --no-headers 2>/dev/null | wc -l
| tr -d ' ')
  if [ "$count" -gt 0 ]; then
    echo -e "${ns}\t${count}"
  fi
done
```

OR show namespaces that do not have Network Policies defined

```
echo -e "NAMESPACES_WITHOUT_NETWORKPOLICIES" && \
for ns in $(kubectl get ns -o jsonpath='{.items[*].metadata.name}'); do
  if ! kubectl get networkpolicy -n "$ns" --no-headers 2>/dev/null | grep -q
.; then
    echo "$ns"
  fi
done
```

Ensure that each namespace defined in the cluster has at least one Network Policy.

Remediation:

Follow the documentation and create `NetworkPolicy` objects as you need them.

Clusters you create with Container Engine for Kubernetes have flannel installed as the default CNI network provider. flannel is a simple overlay virtual network that satisfies the requirements of the Kubernetes networking model by attaching IP addresses to containers.

Although flannel satisfies the requirements of the Kubernetes networking model, it does not support NetworkPolicy resources. If you want to enhance the security of clusters you create with Container Engine for Kubernetes by implementing network policies, you have to install and configure a network provider that does support NetworkPolicy resources. One such provider is Calico (refer to the Kubernetes documentation for a list of other network providers). Calico is an open source networking and network security solution for containers, virtual machines, and native host-based workloads.

Use the Calico open-source software in conjunction with flannel. The Calico Enterprise does not support flannel.

Default Value:

By default, network policies are not created.

References:

1. https://kubernetes.io/docs/concepts/services-networking/networkpolicies/
2. https://octetz.com/posts/k8s-network-policy-apis
3. https://kubernetes.io/docs/tasks/configure-pod-container/declare-network-policy/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●

Controls Version	Control	IG 1	IG 2	IG 3
v7	14.1 Segment the Network Based on Sensitivity Segment the network based on the label or classification level of the information stored on the servers, locate all sensitive information on separated Virtual Local Area Networks (VLANs).		●	●
v7	14.2 Enable Firewall Filtering Between VLANs Enable firewall filtering between VLANs to ensure that only authorized systems are able to communicate with other systems necessary to fulfill their specific responsibilities.		●	●
v6	14.1 Implement Network Segmentation Based On Information Class Segment the network based on the label or classification level of the information stored on the servers. Locate all sensitive information on separated VLANS with firewall filtering to ensure that only authorized individuals are only able to communicate with systems necessary to fulfill their specific responsibilities.			

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1046	TA0007	M1030, M1042

4.4 Secrets Management

4.4.1 Prefer using secrets as files over secrets as environment variables (Automated)

Profile Applicability:

- Level 1

Description:

Kubernetes supports mounting secrets as data volumes or as environment variables. Minimize the use of environment variable secrets.

Rationale:

It is reasonably common for application code to log out its environment (particularly in the event of an error). This will include any secret values passed in as environment variables, so secrets can easily be exposed to any user or entity who has access to the logs.

Impact:

Application code which expects to read secrets in the form of environment variables would need modification

Audit:

Run the following command to find references to objects which use environment variables defined from secrets.

```
kubectl get all -o jsonpath='{range .items[?(@..secretKeyRef)]} {.kind}
{.metadata.name} {"\n"}{end}' -A
```

OR to specifically detect any Pods, Deployments, DaemonSets, etc., that reference environment variables loaded from Kubernetes Secrets (via env[].valueFrom.secretKeyRef):

```
echo -e "NAMESPACE\tKIND\tNAME\tCONTAINER\tSECRET_REF" && \
kubectl get deploy,ds,sts,job,cronjob,pod -A -o json \
| jq -r '
  .items[]
  | .metadata.namespace as $ns
  | .kind as $kind
  | .metadata.name as $name
  | (.spec.template.spec.containers[]?, .spec.containers[]?) as $c
  | ($c.env[]? | select(.valueFrom.secretKeyRef?)) as $env
  | [$ns, $kind, $name, $c.name, $env.valueFrom.secretKeyRef.name] | @tsv
'
```

- -A → searches all namespaces.
- Scans Deployment, DaemonSet, StatefulSet, Job, CronJob, and Pod kinds.
- Looks inside each container's environment variables (env[]) for valueFrom.secretKeyRef.

- Outputs: NAMESPACE | KIND | NAME | CONTAINER | SECRET_REF.
- Works seamlessly in OKE or any standard Kubernetes cluster.

Remediation:

If possible, rewrite application code to read secrets from mounted secret files, rather than from environment variables.

Default Value:

By default, secrets are not defined

References:

1. https://kubernetes.io/docs/concepts/configuration/secret/#using-secrets

Additional Information:

Mounting secrets as volumes has the additional benefit that secret values can be updated without restarting the pod

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.		●	●
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1026

4.4.2 Consider external secret storage (Manual)

Profile Applicability:

- Level 1

Description:

Consider the use of an external secrets storage and management system, instead of using Kubernetes Secrets directly, if you have more complex secret management needs. Ensure the solution requires authentication to access secrets, has auditing of access to and use of secrets, and encrypts secrets. Some solutions also make it easier to rotate secrets.

Rationale:

Kubernetes supports secrets as first-class objects, but care needs to be taken to ensure that access to secrets is carefully limited. Using an external secrets provider can ease the management of access to secrets, especially where secrests are used across both Kubernetes and non-Kubernetes environments.

Impact:

None

Audit:

Review your secrets management implementation.

Remediation:

Refer to the secrets management options offered by your cloud provider or a third-party secrets management solution.

The master nodes in a Kubernetes cluster store sensitive configuration data (such as authentication tokens, passwords, and SSH keys) as Kubernetes secret objects in etcd. Etcd is an open source distributed key-value store that Kubernetes uses for cluster coordination and state management. In the Kubernetes clusters created by Container Engine for Kubernetes, etcd writes and reads data to and from block storage volumes in the Oracle Cloud Infrastructure Block Volume service. Although the data in block storage volumes is encrypted, Kubernetes secrets at rest in etcd itself are not encrypted by default.

For additional security, when you create a new cluster you can specify that Kubernetes secrets at rest in etcd are to be encrypted using the Oracle Cloud Infrastructure Vault service.

Default Value:

By default, no external secret management is configured.

References:

1. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1026

4.5 General Policies

These policies relate to general cluster management topics, like namespace best practices and policies applied to pod objects in the cluster.

4.5.1 Create administrative boundaries between resources using namespaces (Manual)

Profile Applicability:

- Level 1

Description:

Use namespaces to isolate your Kubernetes objects.

Rationale:

Limiting the scope of user permissions can reduce the impact of mistakes or malicious activities. A Kubernetes namespace allows you to partition created resources into logically named groups. Resources created in one namespace can be hidden from other namespaces. By default, each resource created by a user in Kubernetes cluster runs in a default namespace, called `default`. You can create additional namespaces and attach resources and users to them. You can use Kubernetes Authorization plugins to create policies that segregate access to namespace resources between different users.

Impact:

You need to switch between namespaces for administration.

Audit:

Run the below command and review the namespaces created in the cluster.

```
kubectl get namespaces
```

Ensure that these namespaces are the ones you need and are adequately administered as per your requirements.

Remediation:

Follow the documentation and create namespaces for objects in your deployment as you need them.

Default Value:

By default, Kubernetes starts with two initial namespaces:

1. `default` - The default namespace for objects with no other namespace
2. `kube-system` - The namespace for objects created by the Kubernetes system
3. `kube-public` - The namespace for public-readable ConfigMap
4. `kube-node-lease` - The namespace for associated lease object for each node

References:

1. https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/
2. http://blog.kubernetes.io/2016/08/security-best-practices-kubernetes-deployment.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			
v6	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1106, T1609	TA0002	M1038

4.5.2 Apply Security Context to Your Pods and Containers (Manual)

Profile Applicability:

- Level 1

Description:

Apply Security Context to Your Pods and Containers

Rationale:

A security context defines the operating system security settings (uid, gid, capabilities, SELinux role, etc..) applied to a container. When designing your containers and pods, make sure that you configure the security context for your pods, containers, and volumes. A security context is a property defined in the deployment yaml. It controls the security parameters that will be assigned to the pod/container/volume. There are two levels of security context: pod level security context, and container level security context.

Impact:

If you incorrectly apply security contexts, you may have trouble running the pods.

Audit:

Review the pod definitions in your cluster and verify that you have security contexts defined as appropriate.

Here is a key compliance check in OKE (Oracle Kubernetes Engine) to ensure all Pods define a securityContext at either the pod or container level:

```
echo -e
"NAMESPACE\tPOD\tHAS_POD_SECURITY_CONTEXT\tHAS_CONTAINER_SECURITY_CONTEXT" &&
\
kubectl get pods -A -o json \
| jq -r '
  .items[]
  | .metadata.namespace as $ns
  | .metadata.name as $pod
  | (.spec.securityContext != null) as $hasPodSC
  | ([(.spec.containers[]?.securityContext != null)] | any) as
$hasContainerSC
  | [$ns, $pod, $hasPodSC, $hasContainerSC] | @tsv
'
```

- HAS_POD_SECURITY_CONTEXT → checks for a pod-level .spec.securityContext.
- HAS_CONTAINER_SECURITY_CONTEXT → verifies if any container in the pod defines .securityContext.

- jq any — returns true if at least one container has a security context.
- Works across all namespaces (-A).

Sample Ouput:

```
NAMESPACE      POD                                  HAS_POD_SECURITY_CONTEXT
       HAS_CONTAINER_SECURITY_CONTEXT
default         file-check                          true
true
kube-system    coredns-bc868476b-g8285                     true
true
kube-system    csi-oci-node-hsr28                  true
true
kube-system    kube-dns-autoscaler-8b86f9c9f-fzqsv  true
true
kube-system    kube-proxy-g949b                    true
true
kube-system    proxymux-client-5p85m               true
true
kube-system    vcn-native-ip-cni-jwxtl                     true
true
dev             api-service-7dd55b8f65               false
false
```

Remediation:

As a best practice we recommend that you scope the binding for privileged pods to service accounts within a particular namespace, e.g. kube-system, and limiting access to that namespace. For all other serviceaccounts/namespaces, we recommend implementing a more restrictive policy such as this:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
    name: restricted
    annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames:
'docker/default,runtime/default'
    apparmor.security.beta.kubernetes.io/allowedProfileNames:
'runtime/default'
    seccomp.security.alpha.kubernetes.io/defaultProfileName:
'runtime/default'
    apparmor.security.beta.kubernetes.io/defaultProfileName:
'runtime/default'
spec:
    privileged: false
    # Required to prevent escalations to root.
    allowPrivilegeEscalation: false
    # This is redundant with non-root + disallow privilege escalation,
    # but we can provide it for defense in depth.
    requiredDropCapabilities:
    - ALL
    # Allow core volume types.
    volumes:
    - 'configMap'
    - 'emptyDir'
    - 'projected'
    - 'secret'
    - 'downwardAPI'
    # Assume that persistentVolumes set up by the cluster admin are safe to
use.
    - 'persistentVolumeClaim'
    hostNetwork: false
    hostIPC: false
    hostPID: false
    runAsUser:
    # Require the container to run without root privileges.
    rule: 'MustRunAsNonRoot'
    seLinux:
    # This policy assumes the nodes are using AppArmor rather than SELinux.
    rule: 'RunAsAny'
    supplementalGroups:
    rule: 'MustRunAs'
    ranges:
        # Forbid adding the root group.
        - min: 1
        max: 65535
    fsGroup:
    rule: 'MustRunAs'
    ranges:
        # Forbid adding the root group.
        - min: 1
        max: 65535
    readOnlyRootFilesystem: false
```

This policy prevents pods from running as privileged or escalating privileges. It also restricts the types of volumes that can be mounted and the root supplemental groups that can be added.

Another, albeit similar, approach is to start with policy that locks everything down and incrementally add exceptions for applications that need looser restrictions such as logging agents which need the ability to mount a host path.

Default Value:

By default, no security contexts are automatically applied to pods.

References:

1. https://kubernetes.io/docs/concepts/policy/security-context/
2. https://learn.cisecurity.org/benchmarks
3. https://aws.github.io/aws-eks-best-practices/pods/#restrict-the-containers-that-can-run-as-privileged
4. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1556, T1611	TA0004, TA0006	M1048

4.5.3 The default namespace should not be used (Automated)

Profile Applicability:

- Level 1

Description:

Kubernetes provides a default namespace, where objects are placed if no namespace is specified for them. Placing objects in this namespace makes application of RBAC and other controls more difficult.

Rationale:

Resources in a Kubernetes cluster should be segregated by namespace, to allow for security controls to be applied at that level and to make it easier to manage resources.

Impact:

None

Audit:

Run this command to list objects in default namespace

```
kubectl get all -n default
```

The only entries there should be system managed resources such as the `kubernetes` service

OR

```
kubectl get pods -n default
```

Returning No resources found in default namespace.

Remediation:

Ensure that namespaces are created to allow for appropriate segregation of Kubernetes resources and that all new resources are created in a specific namespace.

Default Value:

Unless a namespace is specific on object creation, the `default` namespace will be used

References:

1. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.2 Establish and Maintain a Secure Network Architecture Establish and maintain a secure network architecture. A secure network architecture must address segmentation, least privilege, and availability, at a minimum.		●	●
v7	12.7 Deploy Network-Based Intrusion Prevention Systems Deploy network-based Intrusion Prevention Systems (IPS) to block malicious network traffic at each of the organization's network boundaries.			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1578	TA0005	M1018

5 Managed services

This section consists of security recommendations for the Oracle OKE. These recommendations are applicable for configurations that Oracle OKE customers own and manage.

5.1 Image Registry and Image Scanning

This section contains recommendations relating to container image registries and securing images in those registries, such as Oracle Container Registry (OCR).

5.1.1 Oracle Cloud Security Penetration and Vulnerability Testing (Manual)

Profile Applicability:

- Level 1

Description:

Oracle regularly performs penetration and vulnerability testing and security assessments against the Oracle Cloud infrastructure, platforms, and applications. These tests are intended to validate and improve the overall security of Oracle Cloud services.

Rationale:

Vulnerabilities in software packages can be exploited by hackers or malicious users to obtain unauthorized access to local cloud resources. Oracle Cloud Container Analysis and other third party products allow images stored in Oracle Cloud to be scanned for known vulnerabilities.

Impact:

None.

Audit:

As a service administrator, you can run tests for some Oracle Cloud services. Before running the tests, you must first review the Oracle Cloud Testing Policies section. Follow the steps below to notify Oracle of a penetration and vulnerability test.

Remediation:

As a service administrator, you can run tests for some Oracle Cloud services. Before running the tests, you must first review the Oracle Cloud Testing Policies section.

Note:

You must have an Oracle Account with the necessary privileges to file service maintenance requests, and you must be signed in to the environment that will be the subject of the penetration and vulnerability testing.

Submitting a Cloud Security Testing Notification

References:

1. https://docs.cloud.oracle.com/en-us/iaas/Content/Security/Concepts/security_testing-policy.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.6 Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets Perform automated vulnerability scans of externally-exposed enterprise assets using a SCAP-compliant vulnerability scanning tool. Perform scans on a monthly, or more frequent, basis.		●	●
v7	3 Continuous Vulnerability Management Continuous Vulnerability Management			
v7	3.1 Run Automated Vulnerability Scanning Tools Utilize an up-to-date SCAP-compliant vulnerability scanning tool to automatically scan all systems on the network on a weekly or more frequent basis to identify all potential vulnerabilities on the organization's systems.		●	●
v7	3.2 Perform Authenticated Vulnerability Scanning Perform authenticated vulnerability scanning with agents running locally on each system or with remote scanners that are configured with elevated rights on the system being tested.		●	●

5.1.2 Minimize user access control to Container Engine for Kubernetes (Manual)

Profile Applicability:

- Level 1

Description:

Restrict user access to OKE, limiting interaction with build images to only authorized personnel and service accounts.

Rationale:

Weak access control to OKE may allow malicious users to replace built images with vulnerable or backdoored containers.

Impact:

Care should be taken not to remove access to Oracle Cloud Infrastructure Registry (OCR) for accounts that require this for their operation. Any account granted the Storage Object Viewer role at the project level can view all objects stored in OCS for the project.

Audit:

For most operations on Kubernetes clusters created and managed by Container Engine for Kubernetes, Oracle Cloud Infrastructure Identity and Access Management (IAM) provides access control. A user's permissions to access clusters comes from the groups to which they belong. The permissions for a group are defined by policies. Policies define what actions members of a group can perform, and in which compartments. Users can then access clusters and perform operations based on the policies set for the groups they are members of.

IAM provides control over:

- whether a user can create or delete clusters
- whether a user can add, remove, or modify node pools
- which Kubernetes object create/delete/view operations a user can perform on all clusters within a compartment or tenancy

See Policy Configuration for Cluster Creation and Deployment.

In addition to IAM, the Kubernetes RBAC Authorizer can enforce additional fine-grained access control for users on specific clusters via Kubernetes RBAC roles and clusterroles. A Kubernetes RBAC role is a collection of permissions. For example, a role might include read permission on pods and list permission for pods. A Kubernetes RBAC clusterrole is just like a role, but can be used anywhere in the cluster. A Kubernetes RBAC rolebinding maps a role to a user or set of users, granting that role's permissions to those users for resources in that namespace. Similarly, a Kubernetes RBAC clusterrolebinding maps a clusterrole to a user or set of users, granting that clusterrole's permissions to those users across the entire cluster.

Remediation:

By default, users are not assigned any Kubernetes RBAC roles (or clusterroles) by default. So before attempting to create a new role (or clusterrole), you must be assigned an appropriately privileged role (or clusterrole). A number of such roles and clusterroles are always created by default, including the cluster-admin clusterrole (for a full list, see Default Roles and Role Bindings in the Kubernetes documentation). The cluster-admin clusterrole essentially confers super-user privileges. A user granted the cluster-admin clusterrole can perform any operation across all namespaces in a given cluster.

Note that Oracle Cloud Infrastructure tenancy administrators already have sufficient privileges, and do not require the cluster-admin clusterrole.

See: Granting the Kubernetes RBAC cluster-admin clusterrole

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1078, T1078.002	TA0001, TA0004	M1026

5.1.3 Minimize cluster access to read-only (Manual)

Profile Applicability:

- Level 1

Description:

Configure the Cluster Service Account to only allow read-only access to OKE.

Rationale:

The Cluster Service Account does not require administrative access to OCR, only requiring pull access to containers to deploy onto OKE. Restricting permissions follows the principles of least privilege and prevents credentials from being abused beyond the required role.

Impact:

A separate dedicated service account may be required for use by build servers and other robot users pushing or managing container images.

Audit:

Review Oracle OCS worker node IAM role IAM Policy Permissions to verify that they are set and the minimum required level.

If utilizing a 3rd party tool to scan images utilize the minimum required permission level required to interact with the cluster - generally this should be read-only.

Remediation:

To access a cluster using kubectl, you have to set up a Kubernetes configuration file (commonly known as a 'kubeconfig' file) for the cluster. The kubeconfig file (by default named config and stored in the $HOME/.kube directory) provides the necessary details to access the cluster. Having set up the kubeconfig file, you can start using kubectl to manage the cluster.

The steps to follow when setting up the kubeconfig file depend on how you want to access the cluster:

- To access the cluster using kubectl in Cloud Shell, run an Oracle Cloud Infrastructure CLI command in the Cloud Shell window to set up the kubeconfig file.
- To access the cluster using a local installation of kubectl:
    1. Generate an API signing key pair (if you don't already have one).
    2. Upload the public key of the API signing key pair.
    3. Install and configure the Oracle Cloud Infrastructure CLI.
    4. Set up the kubeconfig file.

See Setting Up Local Access to Clusters

Default Value:

The default permissions for the cluster Service account is dependent on the initial configuration and IAM policy.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1026

5.1.4 Minimize Container Registries to only those approved (Manual)

Profile Applicability:

- Level 1

Description:

Limiting OKE clusters to pull images only from approved container registries is a key security control that prevents the deployment of unverified or malicious container images. By explicitly allowing access to trusted registries—such as Oracle Cloud Infrastructure (OCI) Registry (OCIR) or other vetted enterprise repositories—you ensure that all deployed containers come from sources that meet your organization's security, compliance, and provenance standards.

Rationale:

Allowing unrestricted access to external container registries provides the opportunity for malicious or unapproved containers to be deployed into the cluster. Allow listing only approved container registries reduces this risk.

Impact:

All container images to be deployed to the cluster must be hosted within an approved container image registry.

Audit:

Practice Recommendations:

1. Use Only Trusted Registries - Configure workloads to pull images exclusively from approved registries (e.g., iad.ocir.io, phx.ocir.io, or your enterprise repository).
2. Restrict Outbound Egress - Use Kubernetes Network Policies, OCI Network Security Groups (NSGs), or Security Lists to block access to public internet registries (e.g., Docker Hub) unless explicitly required.
3. Implement Image Pull Policies - Enforce imagePullPolicy: Always and deploy Admission Controllers (e.g., OPA Gatekeeper or Kyverno) to validate image sources before pods are admitted to the cluster.
4. Use Image Signing and Scanning - Enable OCIR Vulnerability Scanning or integrate third-party tools to ensure only signed and scanned images are allowed.

Audit Deployed Images regularly and periodically check for images from unapproved registries:

```
echo -e "NAMESPACE\tPOD_NAME\tIMAGE" && \
kubectl get pods -A -o jsonpath='{range
.items[*]}{.metadata.namespace}{"\t"}{.metadata.name}{"\t"}{.spec.containers[
*].image}{"\n"}{end}' \
| grep -vE '(iad\.ocir\.io|phx\.ocir\.io|your-approved-registry\.com)'
```

Remediation:

Minimize Container Registries to Only Approved Sources

- Define Approved Registries - Establish a list of trusted registries (for example, iad.ocir.io//, phx.ocir.io//, or a private enterprise registry).
- Restrict Pod Image Sources via Admission Controls
- Restrict Egress to Non-Approved Registries - Use Kubernetes NetworkPolicies, OCI Network Security Groups (NSGs), or Security Lists to block egress traffic to public registries
- Enforce Image Scanning and Signing - Enable OCI Vulnerability Scanning Service (VSS) for your OCIR repositories and allow only scanned and signed images to be deployed.

Default Value:

Oracle Kubernetes Engine (OKE) clusters do not restrict which container registries can be used for pulling images

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●
v7	5.3 Securely Store Master Images Store the master images and templates on securely configured servers, validated with integrity monitoring tools, to ensure that only authorized changes to the images are possible.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1133, T1195	TA0001, TA0003	M1016, M1042

5.2 Identity and Access Management (IAM)

This section contains recommendations relating to using Oracle Cloud IAM with OKE.

5.2.1 Prefer using dedicated Service Accounts (Automated)

Profile Applicability:

- Level 1

Description:

Kubernetes workloads should not use cluster node service accounts to authenticate to Oracle Cloud APIs. Each Kubernetes Workload that needs to authenticate to other Oracle services using Cloud IAM should be provisioned a dedicated Service account.

Rationale:

Manual approaches for authenticating Kubernetes workloads running on OKE against Oracle Cloud APIs are: storing service account keys as a Kubernetes secret (which introduces manual key rotation and potential for key compromise); or use of the underlying nodes' IAM Service account, which violates the principle of least privilege on a multitenanted node, when one pod needs to have access to a service, but every other pod on the node that uses the Service account does not.

Audit:

For each namespace in the cluster, review the rights assigned to the default service account and ensure that it has no roles or cluster roles bound to it apart from the defaults.

```
kubectl get serviceaccounts/default -o yaml
```

Additionally ensure that the `automountServiceAccountToken: false` setting is in place for each default service account.

Remediation:

When you create a pod, if you do not specify a service account, it is automatically assigned the default service account in the same namespace. If you get the raw json or yaml for a pod you have created `(for example, kubectl get pods/<podname> -o yaml)`, you can see the spec.serviceAccountName field has been automatically set.

See Configure Service Accounts for Pods

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.5 Establish and Maintain an Inventory of Service Accounts Establish and maintain an inventory of service accounts. The inventory, at a minimum, must contain department owner, review date, and purpose. Perform service account reviews to validate that all active accounts are authorized, on a recurring schedule at a minimum quarterly, or more frequently.		●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

5.3 Cloud Key Management Service (Cloud KMS)

This section contains recommendations relating to using Cloud KMS with OKE.

5.3.1 Encrypting Kubernetes Secrets at Rest in Etcd (Manual)

Profile Applicability:

- Level 1

Description:

Encrypt Kubernetes secrets, stored in etcd, at the application-layer using a customer-managed key.

Rationale:

The master nodes in a Kubernetes cluster store sensitive configuration data (such as authentication tokens, passwords, and SSH keys) as Kubernetes secret objects in etcd. Etcd is an open source distributed key-value store that Kubernetes uses for cluster coordination and state management. In the Kubernetes clusters created by Container Engine for Kubernetes, etcd writes and reads data to and from block storage volumes in the Oracle Cloud Infrastructure Block Volume service. Although the data in block storage volumes is encrypted, Kubernetes secrets at rest in etcd itself are not encrypted by default.

Audit:

Before you can create a cluster where Kubernetes secrets are encrypted in the etcd key-value store, you have to:

- know the name and OCID of a suitable master encryption key in Vault
- create a dynamic group that includes all clusters in the compartment in which you are going to create the new cluster
- create a policy authorizing the dynamic group to use the master encryption key

Remediation:

You can create a cluster in one tenancy that uses a master encryption key in a different tenancy. In this case, you have to write cross-tenancy policies to enable the cluster in its tenancy to access the master encryption key in the Vault service's tenancy. Note that if you want to create a cluster and specify a master encryption key that's in a different tenancy, you cannot use the Console to create the cluster.

For example, assume the cluster is in the ClusterTenancy, and the master encryption key is in the KeyTenancy. Users belonging to a group (OKEAdminGroup) in the ClusterTenancy have permissions to create clusters. A dynamic group (OKEAdminDynGroup) has been created in the cluster, with the rule ALL `{resource.type = 'cluster', resource.compartment.id = 'ocid1.compartment.oc1..<unique_ID>'}`, so all clusters created in the ClusterTenancy belong to the dynamic group.

In the root compartment of the KeyTenancy, the following policies:

- use the ClusterTenancy's OCID to map ClusterTenancy to the alias OKE_Tenancy
- use the OCIDs of OKEAdminGroup and OKEAdminDynGroup to map them to the aliases RemoteOKEAdminGroup and RemoteOKEClusterDynGroup respectively
- give RemoteOKEAdminGroup and RemoteOKEClusterDynGroup the ability to list, view, and perform cryptographic operations with a particular master key in the KeyTenancy

```
Define tenancy OKE_Tenancy as ocid1.tenancy.oc1..<unique_ID>
Define dynamic-group RemoteOKEClusterDynGroup as
ocid1.dynamicgroup.oc1..<unique_ID>
Define group RemoteOKEAdminGroup as ocid1.group.oc1..<unique_ID>
Admit dynamic-group RemoteOKEClusterDynGroup of tenancy ClusterTenancy to use
keys in tenancy where target.key.id = 'ocid1.key.oc1..<unique_ID>'
Admit group RemoteOKEAdminGroup of tenancy ClusterTenancy to use keys in
tenancy where target.key.id = 'ocid1.key.oc1..<unique_ID>'
```

In the root compartment of the ClusterTenancy, the following policies:

- use the KeyTenancy's OCID to map KeyTenancy to the alias KMS_Tenancy
- give OKEAdminGroup and OKEAdminDynGroup the ability to use master keys in the KeyTenancy
- allow OKEAdminDynGroup to use a specific master key obtained from the KeyTenancy in the ClusterTenancy

```
Define tenancy KMS_Tenancy as ocid1.tenancy.oc1..<unique_ID>
Endorse group OKEAdminGroup to use keys in tenancy KMS_Tenancy
Endorse dynamic-group OKEAdminDynGroup to use keys in tenancy KMS_Tenancy
Allow dynamic-group OKEAdminDynGroup to use keys in tenancy where
target.key.id = 'ocid1.key.oc1..<unique_ID>'
```

See Accessing Object Storage Resources Across Tenancies for more examples of writing cross-tenancy policies.

Having entered the policies, you can now run a command similar to the following to create a cluster in the ClusterTenancy that uses the master key obtained from the KeyTenancy:

```
oci ce cluster create --name oke-with-cross-kms --kubernetes-version v1.16.8
--vcn-id ocid1.vcn.oc1.iad.<unique_ID> --service-lb-subnet-ids
'["ocid1.subnet.oc1.iad.<unique_ID>"]' --compartment-id
ocid1.compartment.oc1..<unique_ID> --kms-key-id ocid1.key.oc1.iad.<unique_ID>
```

References:

1. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Tasks/contengencryptingdata.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.		●	●
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1552	TA0006	M1022

5.4 Cluster Networking

This section contains recommendations relating to network security configurations in OKE.

5.4.1 Restrict Access to the Control Plane Endpoint (Automated)

Profile Applicability:

- Level 1

Description:

Enable Authorized IP CIDRs `endpointConfig.authorizedIpCidrs`, used together with `isPublicIpEnabled` to restrict or disable control-plane access. These settings define which IP CIDRs can reach the Kubernetes API server (control plane endpoint).

Rationale:

Restricting access to the Oracle Kubernetes Engine (OKE) cluster control plane through the Authorized IP CIDRs list (`endpointConfig.authorizedIpCidrs`), in combination with the `isPublicIpEnabled` setting, is a critical security control that ensures only trusted administrative networks or systems can reach the Kubernetes API server.

By default, enabling a public control plane endpoint without restrictions exposes the cluster's management interface to the internet, increasing the risk of unauthorized access, brute-force attacks, or exploitation of exposed API vulnerabilities. The Authorized IP CIDRs list acts as a network-level allowlist, explicitly defining which external IP address ranges are permitted to communicate with the cluster's API server.

When used together with `isPublicIpEnabled=false`, this configuration can completely isolate the control plane within the Virtual Cloud Network (VCN), allowing only internal traffic or secure connections via bastion hosts or VPNs. In cases where `isPublicIpEnabled=true` is required for operational purposes, defining a precise Authorized IP CIDR range ensures that only approved administrators or automation systems can manage the cluster remotely.

Impact:

Failure to implement the Authorized IP CIDRs list in conjunction with `isPublicIpEnabled` exposes the OKE cluster control plane to unauthorized network access. This increases the risk of external attacks, including credential theft, denial-of-service attempts, or exploitation of API vulnerabilities. Conversely, enforcing these settings significantly reduces the cluster's attack surface, ensuring that only trusted networks and administrators can interact with the Kubernetes API, thereby maintaining the integrity and confidentiality of the cluster's management plane.

Audit:

Below is a command-line audit procedure to verify that each OKE cluster control plane (master endpoint) is properly restricted using the Authorized IP CIDRs list (`endpointConfig.authorizedIpCidrs`) and the `isPublicIpEnabled` setting.

```
echo -e "NAME\tPUBLIC\tAUTHORIZED_CIDRS" && \
oci ce cluster list --compartment-id "${COMPARTMENT_ID}" --all --output json
\
| jq -r '
  .data[] |
  . as $c |
  ($c.endpointConfig.isPublicIpEnabled // false) as $pub |
  ($c.endpointConfig.authorizedIpCidrs // []) as $cidrs |
  [$c.name, $pub, ($cidrs | join(","))] | @tsv
'
```

Or check the following:

```
export CLUSTER_ID=<oke cluster id>
oci ce cluster get --cluster-id $CLUSTER_ID
```

Check for endpoint URL of the Kubernetes API Server in output

Remediation:

If the OKE cluster control plane is publicly accessible without an authorized IP allowlist, update the endpoint configuration to either disable public access or restrict it to specific trusted IP ranges. This ensures only approved administrative networks can reach the Kubernetes API server, reducing exposure to unauthorized access.

Restrict to Authorized IPs:

```
oci ce cluster update-endpoint-config \
  --cluster-id "${CLUSTER_ID}" \
  --is-public-ip-enabled true \
  --authorized-ip-cidrs '["203.0.113.10/32","198.51.100.0/24"]' \
  --force
```

or make endpoint private only:

```
oci ce cluster update-endpoint-config \
  --cluster-id "${CLUSTER_ID}" \
  --is-public-ip-enabled false \
  --force
```

Default Value:

`isPublicIpEnabled` is defaulted to true and `authorizedIpCidrs` is defaulted to [] (empty list)

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	9.3 Maintain and Enforce Network-Based URL Filters Enforce and update network-based URL filters to limit an enterprise asset from connecting to potentially malicious or unapproved websites. Example implementations include category-based filtering, reputation-based filtering, or through the use of block lists. Enforce filters for all enterprise assets.		●	●

Controls Version	Control	IG 1	IG 2	IG 3
v7	7.5 Subscribe to URL-Categorization service Subscribe to URL categorization services to ensure that they are up-to-date with the most recent website category definitions available. Uncategorized sites shall be blocked by default.		●	●

5.4.2 Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Automated)

Profile Applicability:

- Level 1

Description:

Creating Oracle Kubernetes Engine (OKE) clusters with Private Endpoint Enabled and Public Access Disabled ensures that the Kubernetes control plane (API server) is accessible only within the Oracle Cloud Infrastructure (OCI) Virtual Cloud Network (VCN). This configuration isolates cluster management traffic from the public internet, eliminating exposure to unauthorized external access. By keeping the API endpoint private, administrators and automation systems must connect securely through approved internal networks, VPNs, or bastion hosts. This practice aligns with Oracle and CIS Kubernetes security benchmarks, reducing attack surface and strengthening the overall security posture of the OKE environment.

Rationale:

Enabling a private endpoint and disabling public access for OKE clusters ensures that the Kubernetes control plane is accessible only from within trusted Oracle Cloud Infrastructure (OCI) networks. This prevents unauthorized internet-based access to the cluster's API server, reducing exposure to external threats such as brute-force attacks or exploitation of API vulnerabilities. Restricting control-plane communication to private networks enforces network segmentation, aligns with Zero Trust and CIS Kubernetes Benchmark best practices, and significantly enhances the security and integrity of the cluster's management plane.

Impact:

Failure to configure OKE clusters with Private Endpoint Enabled and Public Access Disabled exposes the Kubernetes control plane to the public internet, increasing the risk of unauthorized access, data breaches, and control-plane compromise. Restricting control-plane access to private networks is critical to maintaining a secure, compliant, and resilient Kubernetes environment.

1. Unauthorized Access Risk: Publicly exposed API endpoints can be discovered and targeted by attackers attempting to exploit misconfigurations or weak credentials.
2. Expanded Attack Surface: Allowing internet access to the control plane increases vulnerability to brute-force, denial-of-service, and reconnaissance attacks.
3. Regulatory Noncompliance: Public exposure of administrative interfaces can violate security and compliance frameworks such as ISO 27001.
4. Data and Service Disruption: A compromised control plane can lead to unauthorized cluster modifications, service outages, or data loss.

Audit:

Below is a command that lists both Private Endpoint = Enabled and Public Access = Disabled for all OKE clusters in a given compartment.

```
echo -e "NAME\tPRIVATE_ENDPOINT_ENABLED\tPUBLIC_ACCESS_DISABLED" && \
oci ce cluster list --compartment-id "${COMPARTMENT_ID}" --all --output json
\
| jq -r '
  .data[] |
  . as $c |
  ($c.endpointConfig.isPublicIpEnabled // false) as $public |
  ($public | not) as $private |
  [
    $c.name,
    ($private | tostring),
    (( $public | not ) | tostring)
  ] | @tsv
'
```

or check for the following:

```
oci ce cluster get --cluster-id "${CLUSTER_ID}" --query 'data."endpoint-
config"."is-public-ip-enabled"' --output json
```

Output shows either:

- `false` - Private endpoint enabled (no public access).
- `true` - Public access enabled (private endpoint disabled).

Remediation:

If an OKE cluster is configured with a public control plane endpoint, it should be disabled to reduce exposure and prevent unauthorized access from the internet. Configuring the cluster with a private endpoint ensures that the Kubernetes API server is accessible only within the Oracle Cloud Infrastructure (OCI) Virtual Cloud Network (VCN).

Update statement to disable the public control plane endpoint:

```
oci ce cluster update-endpoint-config \
  --cluster-id "${CLUSTER_ID}" \
  --is-public-ip-enabled false \
  --force
```

The OKE control plane endpoint is now private and no longer reachable from the public internet, ensuring that administrative traffic stays within trusted network boundaries.

Default Value:

By default, the Public Endpoint is disabled.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	9.3 Maintain and Enforce Network-Based URL Filters Enforce and update network-based URL filters to limit an enterprise asset from connecting to potentially malicious or unapproved websites. Example implementations include category-based filtering, reputation-based filtering, or through the use of block lists. Enforce filters for all enterprise assets.		●	●
v7	7.4 Maintain and Enforce Network-Based URL Filters Enforce network-based URL filters that limit a system's ability to connect to websites not approved by the organization. This filtering shall be enforced for each of the organization's systems, whether they are physically at an organization's facilities or not.		●	●

5.4.3 Ensure clusters are created with Private Nodes (Automated)

Profile Applicability:

- Level 1

Description:

Creating Oracle Kubernetes Engine (OKE) clusters with Private Nodes ensures that worker nodes do not have public IP addresses and communicate only within the Oracle Cloud Infrastructure (OCI) Virtual Cloud Network (VCN). This configuration isolates all node-level traffic from the public internet, significantly reducing the risk of external attacks and unauthorized access. Private nodes enhance security by enforcing internal-only communication between workloads, control plane, and OCI services—while still allowing secure outbound access through NAT gateways when needed.

Disable public IP addresses for cluster nodes, so that they only have private IP addresses. Private Nodes are nodes with no public IP addresses.

Rationale:

Disabling public IP addresses on cluster nodes restricts access to only internal networks, forcing attackers to obtain local network access before attempting to compromise the underlying Kubernetes hosts.

Impact:

To enable Private Nodes, the cluster has to also be configured with a private master IP range and IP Aliasing enabled.

Private Nodes do not have outbound access to the public internet. If you want to provide outbound Internet access for your private nodes, you can use Cloud NAT or you can manage your own NAT gateway.

Audit:

Below is a command line statement to list OKE clusters whose node pools that use private nodes (i.e., nodes without public IPs):

```
echo -e "CLUSTER_ID\tNODE_POOL_NAME\tPRIVATE_NODES_ENABLED" && \
oci ce node-pool list --compartment-id "${COMPARTMENT_ID}" --all --output
json \
| jq -r '
  .data[]? as $np
  | ($np.clusterId // $np.cluster_id // $np."cluster-id" // "-") as $cid
  | ($np.nodeConfigDetails.isNodePublicIpEnabled // false) as $pub
  | [$cid, ($np.name // "-"), (( $pub | not ) | tostring)] | @tsv
'
```

Sample Output:

```
CLUSTER_ID                      NODE_POOL_NAME      PRIVATE_NODES_ENABLED
ocid1.cluster.oc1.phx.123       nodepool-prod       true
ocid1.cluster.oc1.phx.1234      dev-pool-01         false
```

Remediation:

If a node pool in your OKE cluster is configured with public IP addresses, update it to use private nodes to prevent direct internet exposure and ensure all traffic stays within the OCI Virtual Cloud Network (VCN).

```
oci ce node-pool update \
  --node-pool-id "${NODEPOOL_ID}" \
  --is-node-public-ip-enabled false \
  --force
```

Default Value:

By default, Private Nodes are disabled.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●
v7	12 Boundary Defense Boundary Defense			

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1556, T1611	TA0004, TA0006	M1048

5.4.4 Ensure Network Policy is Enabled and set as appropriate (Manual)

Profile Applicability:

- Level 1

Description:

Implementing Kubernetes Network Policies in an Oracle Kubernetes Engine (OKE) cluster is a critical security measure for controlling and segregating pod-to-pod communication. By default, all pods in a Kubernetes cluster can freely communicate with each other, which increases the risk of lateral movement if a single workload is compromised. Enforcing network policies allows administrators to explicitly define which pods or namespaces can communicate, ensuring that workloads are properly isolated based on function, sensitivity, or compliance requirements. This segregation helps contain potential security breaches, limits unauthorized data access between applications, and supports the principle of least privilege in network communication—strengthening the overall security posture and reducing the attack surface within the OKE environment.

Configure a Network Policy to restrict pod-to-pod traffic within a cluster and segregate workloads.

Rationale:

By default, all pod to pod traffic within a cluster is allowed. Network Policy creates a pod-level firewall that can be used to restrict traffic between sources. Pod traffic is restricted by having a Network Policy that selects it (through the use of labels). Once there is any Network Policy in a namespace selecting a particular pod, that pod will reject any connections that are not allowed by any Network Policy. Other pods in the namespace that are not selected by any Network Policy will continue to accept all traffic.

Network Policies are managed via the Kubernetes Network Policy API and enforced by a network plugin, simply creating the resource without a compatible network plugin to implement it will have no effect. OKE supports Network Policy enforcement through the use of Calico.

Impact:

Network Policy requires the Network Policy add-on. This add-on is included automatically when a cluster with Network Policy is created, but for an existing cluster, needs to be added prior to enabling Network Policy.

Enabling/Disabling Network Policy causes a rolling update of all cluster nodes, similar to performing a cluster upgrade. This operation is long-running and will block other operations on the cluster (including delete) until it has run to completion.

If Network Policy is used, a cluster must have at least 2 nodes of type `n1-standard-1` or higher. The recommended minimum size cluster to run Network Policy enforcement is 3 `n1-standard-1` instances.

Enabling Network Policy enforcement consumes additional resources in nodes. Specifically, it increases the memory footprint of the `kube-system` process by approximately 128MB, and requires approximately 300 millicores of CPU.

Audit:

Below is a command line statement to check whether Network Policies are configured in your Oracle Kubernetes Engine (OKE) cluster:

```
echo -e "NAMESPACE\tNETWORK_POLICY_COUNT" && \
for ns in $(kubectl get ns -o jsonpath='{.items[*].metadata.name}'); do
  count=$(kubectl get networkpolicy -n "$ns" --no-headers 2>/dev/null | wc -l
| tr -d ' ')
  echo -e "${ns}\t${count}"
done
```

Sample Output

```
NAMESPACE              NETWORK_POLICY_COUNT
default                2
kube-node-lease        3
kube-public            5
kube-system            0
```

- A count of 0 means no NetworkPolicy objects exist in that namespace and pod-to-pod communication is unrestricted.
- A count greater than 0 means network policies are defined and applied, helping isolate and protect workloads.

Remediation:

When creating a new OKE cluster enable network policies by setting the following flag:

For a New OKE Cluster enable Network Policy on Cluster Creation:

```
oci ce cluster create \
  --compartment-id "${COMPARTMENT_ID}" \
  --name "secure-cluster" \
  --vcn-id "${VCN_OCID}" \
  --kubernetes-version "v1.30.1" \
  --is-kubernetes-dashboard-enabled false \
  --options '{"serviceLbSubnetIds":["<subnet_ocid>"],
"isNetworkPolicyEnabled": true}'
```

- Define Namespace-Level Network Policies after cluster creation, apply a restrictive default policy to each namespace
- Add Allow Rules for Required Traffic and create specific NetworkPolicy resources to allow communication only between approved pods or namespaces.

For an Existing OKE Cluster, enable Network Policy on the Existing Cluster:

```
oci ce cluster update \
  --cluster-id "${CLUSTER_ID}" \
  --options '{"isNetworkPolicyEnabled": true}' \
  --force
```

- Apply a Default Deny-All Policy
- Add Fine-Grained Policies for Authorized Communication by defining NetworkPolicy manifests that explicitly allow required traffic between trusted workloads, services, or namespaces.

Default Value:

By default, Network Policy is disabled.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●
v7	9.4 Apply Host-based Firewalls or Port Filtering Apply host-based firewalls or port filtering tools on end systems, with a default-deny rule that drops all traffic except those services and ports that are explicitly allowed.	●	●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1046	TA0007	M1030, M1042

5.4.5 Encrypt traffic to HTTPS load balancers with TLS certificates (Manual)

Profile Applicability:

- Level 1

Description:

Encrypt traffic to HTTPS load balancers using TLS certificates.

Rationale:

Encrypting traffic between users and your Kubernetes workload is fundamental to protecting data sent over the web.

Audit:

Your load balancer vendor can provide details on auditing the certificates and policies required to utilize TLS.

Remediation:

Your load balancer vendor can provide details on configuring HTTPS with TLS.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).		●	●
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.		●	●

MITRE ATT&CK Mappings:

Techniques / Sub-techniques	Tactics	Mitigations
T1609	TA0002	M1035

5.5 Authentication and Authorization

This section contains recommendations relating to authentication and authorization in OKE.

5.5.1 Access Control and Container Engine for Kubernetes (Manual)

Profile Applicability:

- Level 1

Description:

Cluster Administrators should leverage Oracle Groups and Cloud IAM to assign Kubernetes user roles to a collection of users, instead of to individual emails using only Cloud IAM.

Rationale:

For most operations on Kubernetes clusters created and managed by Container Engine for Kubernetes, Oracle Cloud Infrastructure Identity and Access Management (IAM) provides access control. A user's permissions to access clusters comes from the groups to which they belong. The permissions for a group are defined by policies. Policies define what actions members of a group can perform, and in which compartments. Users can then access clusters and perform operations based on the policies set for the groups they are members of.

IAM provides control over:

- whether a user can create or delete clusters
- whether a user can add, remove, or modify node pools
- which Kubernetes object create/delete/view operations a user can perform on all clusters within a compartment or tenancy

See Policy Configuration for Cluster Creation and Deployment

Impact:

Users must now be assigned to the IAM group created to use this namespace and deploy applications. If they are not they will not be able to access the namespace or deploy.

Audit:

By default, users are not assigned any Kubernetes RBAC roles (or clusterroles) by default. So before attempting to create a new role (or clusterrole), you must be assigned an appropriately privileged role (or clusterrole). A number of such roles and clusterroles are always created by default, including the cluster-admin clusterrole (for a full list, see Default Roles and Role Bindings in the Kubernetes documentation). The cluster-admin clusterrole essentially confers super-user privileges. A user granted the cluster-admin clusterrole can perform any operation across all namespaces in a given cluster.

Remediation:

Example: Granting the Kubernetes RBAC cluster-admin clusterrole

Follow these steps to grant a user who is not a tenancy administrator the Kubernetes RBAC cluster-admin clusterrole on a cluster deployed on Oracle Cloud Infrastructure:

1. If you haven't already done so, follow the steps to set up the cluster's kubeconfig configuration file and (if necessary) set the KUBECONFIG environment variable to point to the file. Note that you must set up your own kubeconfig file. You cannot access a cluster using a kubeconfig file that a different user set up. See Setting Up Cluster Access.
2. In a terminal window, grant the Kubernetes RBAC cluster-admin clusterrole to the user by entering:

```
$ kubectl create clusterrolebinding <my-cluster-admin-binding> --
clusterrole=cluster-admin --user=<user_OCID>
```

where:

- is a string of your choice to be used as the name for the binding between the user and the Kubernetes RBAC cluster-admin clusterrole. For example, jdoe_clst_adm
- <user_OCID> is the user's OCID (obtained from the Console). For example, ocid1.user.oc1..aaaaa...zutq (abbreviated for readability).

For example:

```
$ kubectl create clusterrolebinding jdoe_clst_adm --clusterrole=cluster-admin
--user=ocid1.user.oc1..aaaaa...zutq
```

References:

1. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengaboutaccesscontrol.htm

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.5 Centralize Network Authentication, Authorization, and Auditing (AAA) Centralize network AAA.		●	●
v7	16.2 Configure Centralized Point of Authentication Configure access for all accounts through as few centralized points of authentication as possible, including network, security, and cloud systems.		●	●

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**Control Plane Components**		
2	**Control Plane Configuration**		
2.1	**Authentication and Authorization**		
2.1.1	Client certificate authentication should not be used for users (Manual)	☐	☐
2.2	**Logging**		
2.2.1	Ensure access to OCI Audit service Log for OKE (Manual)	☐	☐
3	**Worker Nodes**		
3.1	**Worker Node Configuration Files**		
3.1.1	Ensure that the kubelet-config.json file permissions are set to 644 or more restrictive (Automated)	☐	☐
3.1.2	Ensure that the kubelet-config.json file ownership is set to root:root (Automated)	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Automated)	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root (Automated)	☐	☐
3.2	**Kubelet**		
3.2.1	Ensure that the --anonymous-auth argument is set to false (Automated)	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
3.2.3	Ensure that the --client-ca-file argument is set as appropriate (Automated)	☐	☐
3.2.4	Ensure that the --read-only-port argument is set to 0 (Manual)	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated)	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true (Automated)	☐	☐
3.2.7	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Automated)	☐	☐
3.2.8	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	☐	☐
3.2.9	Ensure that the --rotate-certificates argument is not set to false (Automated)	☐	☐
3.2.10	Ensure that the --rotate-server-certificates argument is set to true (Automated)	☐	☐
4	**Policies**		
4.1	**RBAC and Service Accounts**		
4.1.1	Ensure that the cluster-admin role is only used where required (Manual)	☐	☐
4.1.2	Minimize access to secrets (Manual)	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles (Manual)	☐	☐
4.1.4	Minimize access to create pods (Manual)	☐	☐
4.1.5	Ensure that default service accounts are not actively used. (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.1.6	Ensure that Service Account Tokens are only mounted where necessary (Automated)	☐	☐
4.2	**Pod Security Policies**		
4.2.1	Minimize the admission of privileged containers (Automated)	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace (Automated)	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace (Automated)	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace (Automated)	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation (Automated)	☐	☐
4.3	**CNI Plugin**		
4.3.1	Ensure latest CNI version is used (Automated)	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined (Manual)	☐	☐
4.4	**Secrets Management**		
4.4.1	Prefer using secrets as files over secrets as environment variables (Automated)	☐	☐
4.4.2	Consider external secret storage (Manual)	☐	☐
4.5	**General Policies**		
4.5.1	Create administrative boundaries between resources using namespaces (Manual)	☐	☐
4.5.2	Apply Security Context to Your Pods and Containers (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.5.3	The default namespace should not be used (Automated)	☐	☐
5	**Managed services**		
5.1	**Image Registry and Image Scanning**		
5.1.1	Oracle Cloud Security Penetration and Vulnerability Testing (Manual)	☐	☐
5.1.2	Minimize user access control to Container Engine for Kubernetes (Manual)	☐	☐
5.1.3	Minimize cluster access to read-only (Manual)	☐	☐
5.1.4	Minimize Container Registries to only those approved (Manual)	☐	☐
5.2	**Identity and Access Management (IAM)**		
5.2.1	Prefer using dedicated Service Accounts (Automated)	☐	☐
5.3	**Cloud Key Management Service (Cloud KMS)**		
5.3.1	Encrypting Kubernetes Secrets at Rest in Etcd (Manual)	☐	☐
5.4	**Cluster Networking**		
5.4.1	Restrict Access to the Control Plane Endpoint (Automated)	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Automated)	☐	☐
5.4.3	Ensure clusters are created with Private Nodes (Automated)	☐	☐
5.4.4	Ensure Network Policy is Enabled and set as appropriate (Manual)	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.5	**Authentication and Authorization**		
5.5.1	Access Control and Container Engine for Kubernetes (Manual)	☐	☐

Appendix: CIS Controls v7 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	☐	☐
2.2.1	Ensure access to OCI Audit service Log for OKE	☐	☐
3.1.1	Ensure that the kubelet-config.json file permissions are set to 644 or more restrictive	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture	☐	☐
4.1.4	Minimize access to create pods	☐	☐
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure latest CNI version is used	☐	☐
4.5.2	Apply Security Context to Your Pods and Containers	☐	☐
5.1.2	Minimize user access control to Container Engine for Kubernetes	☐	☐
5.2.1	Prefer using dedicated Service Accounts	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐
5.4.4	Ensure Network Policy is Enabled and set as appropriate	☐	☐

Appendix: CIS Controls v7 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	☐	☐
2.2.1	Ensure access to OCI Audit service Log for OKE	☐	☐
3.1.1	Ensure that the kubelet-config.json file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet-config.json file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
3.2.4	Ensure that the --read-only-port argument is set to 0	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture	☐	☐
3.2.8	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
3.2.9	Ensure that the --rotate-certificates argument is not set to false	☐	☐
3.2.10	Ensure that the --rotate-server-certificates argument is set to true	☐	☐
4.1.4	Minimize access to create pods	☐	☐
4.1.5	Ensure that default service accounts are not actively used.	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure latest CNI version is used	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.5.2	Apply Security Context to Your Pods and Containers	☐	☐
5.1.1	Oracle Cloud Security Penetration and Vulnerability Testing	☐	☐
5.1.2	Minimize user access control to Container Engine for Kubernetes	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.2.1	Prefer using dedicated Service Accounts	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐
5.4.4	Ensure Network Policy is Enabled and set as appropriate	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates	☐	☐
5.5.1	Access Control and Container Engine for Kubernetes	☐	☐

Appendix: CIS Controls v7 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	☐	☐
2.2.1	Ensure access to OCI Audit service Log for OKE	☐	☐
3.1.1	Ensure that the kubelet-config.json file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet-config.json file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
3.2.4	Ensure that the --read-only-port argument is set to 0	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture	☐	☐
3.2.8	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
3.2.9	Ensure that the --rotate-certificates argument is not set to false	☐	☐
3.2.10	Ensure that the --rotate-server-certificates argument is set to true	☐	☐
4.1.4	Minimize access to create pods	☐	☐
4.1.5	Ensure that default service accounts are not actively used.	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure latest CNI version is used	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.4.2	Consider external secret storage	☐	☐
4.5.2	Apply Security Context to Your Pods and Containers	☐	☐
4.5.3	The default namespace should not be used	☐	☐
5.1.1	Oracle Cloud Security Penetration and Vulnerability Testing	☐	☐
5.1.2	Minimize user access control to Container Engine for Kubernetes	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.2.1	Prefer using dedicated Service Accounts	☐	☐
5.3.1	Encrypting Kubernetes Secrets at Rest in Etcd	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐
5.4.4	Ensure Network Policy is Enabled and set as appropriate	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates	☐	☐
5.5.1	Access Control and Container Engine for Kubernetes	☐	☐

Appendix: CIS Controls v7 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐

Appendix: CIS Controls v8 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.2.1	Ensure access to OCI Audit service Log for OKE	☐	☐
3.1.1	Ensure that the kubelet-config.json file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet-config.json file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
3.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.7	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure latest CNI version is used	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.4.2	Consider external secret storage	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.5.1	Create administrative boundaries between resources using namespaces	☐	☐
4.5.2	Apply Security Context to Your Pods and Containers	☐	☐
5.1.3	Minimize cluster access to read-only	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐
5.4.4	Ensure Network Policy is Enabled and set as appropriate	☐	☐

Appendix: CIS Controls v8 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	☐	☐
2.2.1	Ensure access to OCI Audit service Log for OKE	☐	☐
3.1.1	Ensure that the kubelet-config.json file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet-config.json file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
3.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
3.2.4	Ensure that the --read-only-port argument is set to 0	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture	☐	☐
3.2.8	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
3.2.9	Ensure that the --rotate-certificates argument is not set to false	☐	☐
3.2.10	Ensure that the --rotate-server-certificates argument is set to true	☐	☐
4.1.2	Minimize access to secrets	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure latest CNI version is used	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.4.2	Consider external secret storage	☐	☐
4.5.1	Create administrative boundaries between resources using namespaces	☐	☐
4.5.2	Apply Security Context to Your Pods and Containers	☐	☐
4.5.3	The default namespace should not be used	☐	☐
5.1.1	Oracle Cloud Security Penetration and Vulnerability Testing	☐	☐
5.1.3	Minimize cluster access to read-only	☐	☐
5.2.1	Prefer using dedicated Service Accounts	☐	☐
5.3.1	Encrypting Kubernetes Secrets at Rest in Etcd	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐
5.4.4	Ensure Network Policy is Enabled and set as appropriate	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates	☐	☐
5.5.1	Access Control and Container Engine for Kubernetes	☐	☐

Appendix: CIS Controls v8 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	☐	☐
2.2.1	Ensure access to OCI Audit service Log for OKE	☐	☐
3.1.1	Ensure that the kubelet-config.json file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet-config.json file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
3.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
3.2.4	Ensure that the --read-only-port argument is set to 0	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture	☐	☐
3.2.8	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
3.2.9	Ensure that the --rotate-certificates argument is not set to false	☐	☐
3.2.10	Ensure that the --rotate-server-certificates argument is set to true	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Minimize access to create pods	☐	☐
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure latest CNI version is used	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.4.2	Consider external secret storage	☐	☐
4.5.1	Create administrative boundaries between resources using namespaces	☐	☐
4.5.2	Apply Security Context to Your Pods and Containers	☐	☐
4.5.3	The default namespace should not be used	☐	☐
5.1.1	Oracle Cloud Security Penetration and Vulnerability Testing	☐	☐
5.1.2	Minimize user access control to Container Engine for Kubernetes	☐	☐
5.1.3	Minimize cluster access to read-only	☐	☐
5.2.1	Prefer using dedicated Service Accounts	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.3.1	Encrypting Kubernetes Secrets at Rest in Etcd	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐
5.4.4	Ensure Network Policy is Enabled and set as appropriate	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates	☐	☐
5.5.1	Access Control and Container Engine for Kubernetes	☐	☐

Appendix: CIS Controls v8 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
5.1.4	Minimize Container Registries to only those approved	☐	☐

Appendix: Change History

Date	Version	Changes for this version
November 1, 2025	1.8.0	Added and tested support for the latest Kubernetes Cluster version 1.34
October 15th, 2025	1.8.0	Added and tested support for the Kubernetes Cluster version 1.33
October 12, 2025	1.8.0	Updated recommendation 5.10.2 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.10.1 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.9.2 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.9.1 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.8.3 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.8.1 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.5.6 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.5.5 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.5.4 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.5.3 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.5.2 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.5.1 audit statement

Date	Version	Changes for this version
October 12, 2025	1.8.0	Updated recommendation 5.4.1 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.3.1 audit statement
October 12, 2025	1.8.0	Updated recommendation 5.2.2 audit statement
October 10th, 2025	1.8.0	Updated recommendation 5.2.1 audit statement
October 7th, 2025	1.8.0	Updated recommendation 4.6.4 audit statement
October 7th, 2025	1.8.0	Updated recommendation 4.3.2 audit statement
October 7th, 2025	1.8.0	Updated recommendation 4.1.4 audit statement
October 5, 2025	1.8.0	Updated recommendation 4.1.5 audit statement
October 5, 2025	1.8.0	Updated recommendation 4.1.3 audit statement
October 5, 2025	1.8.0	Updated recommendation 4.1.2 audit statement
October 5, 2025	1.8.0	Updated recommendation 4.1.1 audit statement
October 1, 2025	1.8.0	Updated recommendation 3.1.4 audit statement
5/1/2025	1.7.0	Tested AAC for OKE clusters built on Kubernetes v1.31, 1.30, 1.29
5/1/2025	1.7.0	Added Support for OKE clusters built on Kubernetes v1.31
4/22/2025	1.7.0	3.1.3 recommendation edited and AAC test script modified
4/20/2025	1.7.0	3.1.2 recommendation edited to reflect config.json

Date	Version	Changes for this version
4/20/2025	1.7.0	3.1.1 recommendation edited to reflect new path and file name for kubelet-config.json
4/14/2025	1.7.0	3.1.1 Edited AAC test script
4/14/2025	1.7.0	2.1.1 changed from an automated test to manual
11/01/2024	1.6.0	Tested AAC for OKE clusters built on Kubernetes v1.30, 1.29, 1.28
11/01/2024	1.6.0	Added Support for OKE clusters built on Kubernetes v1.30, 1.29, 1.28
5/1/2024	1.5.0	3.1.1 Ensure that the kubeconfig file permissions are set to 644 or more restrictive
5/1/2024	1.5.0	3.1.2 Ensure that the proxy kubeconfig file ownership is set to root:root
5/1/2024	1.5.0	3.1.3 Ensure that the kubelet configuration file has permissions set to 644 or more restrictive
5/1/2024	1.5.0	3.1.4 Ensure that the kubelet configuration file ownership is set to root:root
5/1/2024	1.5.0	4.2.1 Minimize the admission of privileged containers
5/1/2024	1.5.0	4.2.2 Minimize the admission of containers wishing to share the host process ID namespace
5/1/2024	1.5.0	4.2.3 Minimize the admission of containers wishing to share the host IPC namespace
5/1/2024	1.5.0	4.2.4 Minimize the admission of containers wishing to share the host network namespace
5/1/2024	1.5.0	4.2.5 Minimize the admission of containers with allowPrivilegeEscalation
5/1/2024	1.5.0	4.3.2 Ensure that all Namespaces have Network Policies defined
5/1/2024	1.5.0	4.5.3 The default namespace should not be used
5/1/2024	1.5.0	5.2.1 Prefer using dedicated Service Accounts
5/1/2024	1.5.0	5.4.1 Restrict Access to the Control Plane Endpoint

Date	Version	Changes for this version
5/1/2024	1.5.0	5.4.4 Ensure Network Policy is Enabled and set as appropriate
5/1/2024	1.5.0	5.4.5 Encrypt traffic to HTTPS load balancers with TLS certificates
4/25/2024	1.5.0	2.1.2 Ensure OKE service level admins are created to manage OKE
4/25/2024	1.5.0	2.2.2 Ensure that the audit policy covers key security concerns
4/25/2024	1.5.0	4.2.6 Minimize the admission of root containers
4/24/2024	1.5.0	4.2.7 Minimize the admission of containers with added capabilities
4/24/2024	1.5.0	4.2.8 Minimize the admission of containers with capabilities assigned
9/23/2023	1.4.0	Updated and tested all AAC automated content against the latest cluster version/s
9/22/2023	1.4.0	Updated recommendations to sync with Kubernetes 1.8
9/01/2023	1.4.0	Ticket 19131 – update pod security policy reference documents and URLs
5/15/2023	1.3.0	Support and AAC for Kubernetes Engine Versions: 1.2.4 &1.2.5 & 1.2.6
4/15/2023	1.3.0	Ticket 18163 – Added guidance and external links to Kubernetes community best practices for rotating certificates.
10/01/2022	1.2.0	Support for latest Kubernetes Engine available 1.2.4
11/02/2022	1.2.0	Updated audit for recommendation 4.1.1
11/02/2022	1.2.0	Updated and tested AAC to support latest kubernetes engine available on the Oracle Cloud