CIS Benchmarks

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

v1.0 - 11-12-2020

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/

Table of Contents

Overview

This document provides prescriptive guidance for running Oracle Kubernetes Engine (OKE) v1.15 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 benchmark was created using a consensus review process comprised of subject matter experts. 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, or examples. Text should be interpreted exactly as presented.
<italic font in brackets>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

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 Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**
- **Level 2**

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:

Editor/s
Gilson Melo, Randall Mowen

Contributor/s
Randall Mowen
Mark Larinde
Logan Kleier
Adao Oliveira Junior
Sherwood Zern
Josh Hammer

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.

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:

Version 7

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.

2.1.2 Ensure OKE service level admins are created to manage OKE (Manual)

Profile Applicability:

- Level 1

Description:

OKE integrates with Oracle Cloud Infrastructure IAM for authentication and authorization, for all interfaces (the Console, SDK or CLI, and REST API). To create, update, and delete clusters and node pools, users must be in groups and those group must have the necessary access. Certain cluster operations created by Container Engine for Kubernetes may require additional permissions granted via a Kubernetes RBAC role or cluster role. One example is the installing the metrics server which requires additional K8s permissions.

The following required policy statements to enable users to use Container Engine for Kubernetes to create, update, and delete clusters and node pools:

```
    Allow group <group-name> to manage instance-family in <compartment-name>

    Allow group <group-name> to use subnets in <compartment-name>

    Allow group <group-name> to read virtual-network-family in <compartment-
name>

    Allow group <group-name> to use vnics in <compartment-name>

    Allow group <group-name> to inspect compartments in <compartment-name>

    Allow group <group-name> to manage cluster-family in <compartment-name>
```

Rationale:

Creating OKE level administrators helps in tightly controlling access to Oracle Cloud Infrastructure (OCI) services to implement the least-privileged security principle.

Audit:

1. In the Console, open the navigation menu. Under Governance and Administration, go to Identity and click Groups. A list of the groups in the compartment you're viewing is displayed.
2. Select the tenancy's root compartment or an individual compartment containing cluster-related resources from the list on the left.
3. Open the policy associated OKE Administrators
4. Ensure it contains the below statements:

```
    Allow group <group-name> to manage instance-family in <compartment-name>
    Allow group <group-name> to use subnets in <compartment-name>
    Allow group <group-name> to read virtual-network-family in <compartment-
name>
    Allow group <group-name> to use vnics in <compartment-name>
    Allow group <group-name> to inspect compartments in <compartment-name>
    Allow group <group-name> to manage cluster-family in <compartment-name>
```

Remediation:

1. In the Console, open the navigation menu. Go to Identity and click Groups.
2. Click Create Group
3. Enter the following:

- Name: A name for the group (for example, acme-dev-team-oke-group) that is unique within the tenancy.
- Description: A friendly description. You can change this later if you want to. Avoid entering confidential information.

4. Open the navigation menu. Go to Identity and click Policies. A list of the policies in the compartment you're viewing is displayed.
5. Select the tenancy's root compartment or an individual compartment containing cluster-related resources from the list on the left.
6. Click Create Policy.
7. Enter the following:

- Name: A name for the policy (for example, acme-dev-team-oke-required-policy) that is unique within the compartment. If you are creating the policy in the tenancy's root compartment, the name must be unique across all policies in your tenancy. You cannot change this later. Avoid entering confidential information.
- Description: A friendly description. You can change this later if you want to. Avoid entering confidential information.
- Statement: The following required policy statements to enable users to use Container Engine for Kubernetes to create, update, and delete clusters and node pools:

```
    Allow group <group-name> to manage instance-family in <compartment-name>
    Allow group <group-name> to use subnets in <compartment-name>
    Allow group <group-name> to read virtual-network-family in <compartment-
name>
    Allow group <group-name> to use vnics in <compartment-name>
    Allow group <group-name> to inspect compartments in <compartment-name>
    Allow group <group-name> to manage cluster-family in <compartment-name>
```

8. Click Create

2.2 Authentication and Authorization

2.2.1 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.

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:

Version 7

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.

2.2.2 Ensure OKE service level admins are created to manage OKE (Manual)

Profile Applicability:

- Level 1

Description:

OKE integrates with Oracle Cloud Infrastructure IAM for authentication and authorization, for all interfaces (the Console, SDK or CLI, and REST API). To create, update, and delete clusters and node pools, users must be in groups and those group must have the necessary access. Certain cluster operations created by Container Engine for Kubernetes may require additional permissions granted via a Kubernetes RBAC role or cluster role. One example is the installing the metrics server which requires additional K8s permissions.

The following required policy statements to enable users to use Container Engine for Kubernetes to create, update, and delete clusters and node pools:

```
    Allow group <group-name> to manage instance-family in <compartment-name>

    Allow group <group-name> to use subnets in <compartment-name>

    Allow group <group-name> to read virtual-network-family in <compartment-
name>

    Allow group <group-name> to use vnics in <compartment-name>

    Allow group <group-name> to inspect compartments in <compartment-name>

    Allow group <group-name> to manage cluster-family in <compartment-name>
```

Rationale:

Creating OKE level administrators helps in tightly controlling access to Oracle Cloud Infrastructure (OCI) services to implement the least-privileged security principle.

Audit:

1. In the Console, open the navigation menu. Under Governance and Administration, go to Identity and click Groups. A list of the groups in the compartment you're viewing is displayed.
2. Select the tenancy's root compartment or an individual compartment containing cluster-related resources from the list on the left.
3. Open the policy associated OKE Administrators
4. Ensure it contains the below statements:

```
    Allow group <group-name> to manage instance-family in <compartment-name>
    Allow group <group-name> to use subnets in <compartment-name>
    Allow group <group-name> to read virtual-network-family in <compartment-
name>
    Allow group <group-name> to use vnics in <compartment-name>
    Allow group <group-name> to inspect compartments in <compartment-name>
    Allow group <group-name> to manage cluster-family in <compartment-name>
```

Remediation:

1. In the Console, open the navigation menu. Go to Identity and click Groups.
2. Click Create Group
3. Enter the following:

- Name: A name for the group (for example, acme-dev-team-oke-group) that is unique within the tenancy.
- Description: A friendly description. You can change this later if you want to. Avoid entering confidential information.

4. Open the navigation menu. Go to Identity and click Policies. A list of the policies in the compartment you're viewing is displayed.
5. Select the tenancy's root compartment or an individual compartment containing cluster-related resources from the list on the left.
6. Click Create Policy.
7. Enter the following:

- Name: A name for the policy (for example, acme-dev-team-oke-required-policy) that is unique within the compartment. If you are creating the policy in the tenancy's root compartment, the name must be unique across all policies in your tenancy. You cannot change this later. Avoid entering confidential information.
- Description: A friendly description. You can change this later if you want to. Avoid entering confidential information.
- Statement: The following required policy statements to enable users to use Container Engine for Kubernetes to create, update, and delete clusters and node pools:

```
    Allow group <group-name> to manage instance-family in <compartment-name>
    Allow group <group-name> to use subnets in <compartment-name>
    Allow group <group-name> to read virtual-network-family in <compartment-
name>
    Allow group <group-name> to use vnics in <compartment-name>
    Allow group <group-name> to inspect compartments in <compartment-name>
    Allow group <group-name> to manage cluster-family in <compartment-name>
```

8. Click Create

2.3 Logging

2.3.1 Ensure access to OCI Audit service Log for OKE (Automated)

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:

1.1.1 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

2.3.2 Ensure that the audit policy covers key security concerns (Manual)

Profile Applicability:

- Level 1
- Level 2

Description:

Ensure that the audit policy created for the cluster covers key security concerns.

Rationale:

Security audit logs should cover access and modification of key resources in the cluster to enable them to form a significant part of a security environment.

Impact:

Increasing audit logging will consume resources on the nodes or other log destination.

Audit:

This control cannot be audited in OKE.

Remediation:

This control cannot be modified in OKE.

Default Value:

See the OKE documentation for the default value.

References:

1. https://github.com/k8scop/k8s-security-dashboard/blob/master/configs/kubernetes/adv-audit.yaml
2. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/#audit-policy

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 kubeconfig file permissions are set to 644 or more restrictive (Manual)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `kubelet` kubeconfig 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 kubeconfig file.

Impact:

None.

Audit:

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 kubeconfig file:

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

The output of the above command should return something similar to `--kubeconfig /etc/kubernetes/kubelet.conf` and `--bootstrap-kubeconfig`

`/etc/kubernetes/bootstrap-kubelet.conf` which is the location of the kubeconfig files. Run this command to obtain the kubeconfig file permissions:

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

The output of the above command gives you the kubeconfig file's permissions.
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 <kubeconfig 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:

Version 6

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.

Version 7

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.

3.1.2 Ensure that the proxy kubeconfig file ownership is set to root:root (Manual)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The kubeconfig 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:

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 kubeconfig file:

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

The output of the above command should return something similar to `--kubeconfig /etc/kubernetes/kubelet.conf` and `--bootstrap-kubeconfig /etc/kubernetes/bootstrap-kubelet.conf` which is the location of the kubeconfig files.
Run this command to obtain the kubeconfig file ownership:

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

The output of the above command gives you the kubeconfig 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 <kubeconfig 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:

Version 6

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.

Version 7

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.

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

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:

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 kubeconfig file:

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

The output of the above command should return something similar to `--kubeconfig /etc/kubernetes/kubelet.conf` and `--bootstrap-kubeconfig /etc/kubernetes/bootstrap-kubelet.conf` which is the location of the kubeconfig files.
Run this command to obtain the kubeconfig file ownership:

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

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

Remediation:

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

```
chmod 644 /var/lib/kubelet/config.yaml
```

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:

Version 6

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.

Version 7

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.

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

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:

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 kubeconfig file:

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

The output of the above command should return something similar to `--kubeconfig /etc/kubernetes/kubelet.conf` and `--bootstrap-kubeconfig /etc/kubernetes/bootstrap-kubelet.conf` which is the location of the kubeconfig files.
Run this command to obtain the kubeconfig file ownership:

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

The output of the above command is the Kubelet config 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:

Version 6

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.

Version 7

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.

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:

Version 6

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

Version 7

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

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:

Version 6

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

Version 7

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

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:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks
All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

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:

Version 6

9.1 Limit Open Ports, Protocols, and Services
Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

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.

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

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:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

3.2.6 Ensure that the --protect-kernel-defaults argument is set to true (Manual)

Profile Applicability:

- Level 1

Description:

Protect tuned kernel parameters from overriding kubelet default kernel parameter values.

Rationale:

Kernel parameters are usually tuned and hardened by the system administrators before putting the systems into production. These parameters protect the kernel and the system. Your kubelet kernel defaults that rely on such parameters should be appropriately set to match the desired secured system state. Ignoring this could potentially lead to running pods with undesired kernel behavior.

Impact:

You would have to re-tune kernel parameters to match kubelet parameters.

Audit:

Audit Method 1:
If using a Kubelet configuration file, check that there is an entry for `--protect-kernel-defaults` 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 `--protect-kernel-defaults=true`.
Audit Method 2:
If using the api configz endpoint consider searching for the status of

`"protectKernelDefaults"` 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

```
--protect-kernel-defaults=true
```

Remediation Method 2:
If using the api configz endpoint consider searching for the status of `"protectKernelDefaults":` 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:

Version 6

3 Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers
Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers

Version 7

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.

3.2.7 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:

Version 6

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

Version 7

9 Limitation and Control of Network Ports, Protocols, and Services
Limitation and Control of Network Ports, Protocols, and Services

3.2.8 Ensure that the --hostname-override argument is not set (Manual)

Profile Applicability:

- Level 1

Description:

Do not override node hostnames.

Rationale:

Overriding hostnames could potentially break TLS setup between the kubelet and the apiserver. Additionally, with overridden hostnames, it becomes increasingly difficult to associate logs with a particular node and process them for security analytics. Hence, you should setup your kubelet nodes with resolvable FQDNs and avoid overriding the hostnames with IPs.

Impact:

Some cloud providers may require this flag to ensure that hostname matches names issued by the cloud provider. In these environments, this recommendation should not apply.

Audit:

Audit Method 1:
If using a Kubelet configuration file, check that there is an entry for `--hostname-override` exists and is set to the Cluster Node Name.
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 `--hostname-override` exists and is set to the Cluster Node Name.

Remediation:

Remediation Method 1:
If modifying the Kubelet config file, edit the kubelet.service file `/etc/systemd/system/kubelet-.service` and set the below parameter

```
--hostname-override=NODE NAME (where NODE NAME is the internal IP ex.
10.0.10.4, as assigned my OKE on build)
```

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/issues/22063
3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Version 6

3 Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers
Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers

Version 7

5 Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers
Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers

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

Profile Applicability:

- Level 2

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:

Version 6

6 Maintenance, Monitoring, and Analysis of Audit Logs
Maintenance, Monitoring, and Analysis of Audit Logs

Version 7

6 Maintenance, Monitoring and Analysis of Audit Logs
Maintenance, Monitoring and Analysis of Audit Logs

3.2.10 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:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks
All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

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

Profile Applicability:

- Level 2

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:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks
All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

3.2.12 Ensure that the --rotate-server-certificates argument is set to true (Manual)

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

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks
All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

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.
kubectl get clusterrolebindings -o=custom-columns=NAME:.metadata.name,ROLE:.roleRef.name,SUBJECT:.subjects[*].name
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:

Version 6

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.

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

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.

Remediation:

Where possible, remove `get`, `list` and `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:

Version 7

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.

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

```
kubectl get roles --all-namespaces -o yaml
```

Retrieve the cluster roles defined in the cluster and review for wildcards

```
kubectl get clusterroles -o yaml
```

Remediation:

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

CIS Controls:

Version 7

5.1 Establish Secure Configurations
Maintain documented, standard security configuration standards for all authorized operating systems and software.

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:

Review the users who have create access to pod objects in the Kubernetes API.

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:clusterrole-aggregation-controller  clusterrole-
aggregation-controller  ServiceAccount  kube-system
```

```
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:

Version 6

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.

Version 7

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.

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

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.

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:

Version 7

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.

5.2 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.

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

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
```

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:

Version 6

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.

Version 7

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.

4.2 Pod Security Policies

A Pod Security Policy (PSP) is a cluster-level resource that controls security settings for pods. Your cluster may have multiple PSPs. You can query PSPs with the following command:

```
kubectl get psp
```

PodSecurityPolicies are used in conjunction with the PodSecurityPolicy admission controller plugin.

4.2.1 Minimize the admission of privileged containers (Manual)

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 PodSecurityPolicy (PSP) defined which does not permit privileged containers.

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

```
kubectl get psp -o json
```

Verify that there is at least one PSP which does not return `true`.
`kubectl get psp <name> -o=jsonpath='{.spec.privileged}'`

Remediation:

Create a PSP as described in the Kubernetes documentation, ensuring that the `.spec.privileged` field is omitted or set to the correct value.

Default Value:

By default, PodSecurityPolicies are not defined.

Before enabling the PodSecurityPolicy admission controller of an existing cluster it is strongly recommended you first verify the cluster's pod security policies in a development or test environment. That way, you can be sure the pod security policies work as you expect and correctly allow (or refuse) pods to start on the cluster. It is recommended to follow the outlined procedure:

1. Create the kubernetes cluster, but do not enable the PodSecurityPolicy admission controller unless you have verified your deployed cluster's pod security policies in a development or test environment.
2. Deploy the pod security policies to the cluster.
3. Access the console and edit the Kubernetes cluster policy from disabled to enabled.

It is very important to note that when you enable a cluster's PodSecurityPolicy admission controller, no application pods can start on the cluster unless suitable pod security policies exist, along with roles (or clusterroles) and rolebindings (or clusterrolebindings) to associate pods with policies. You will not be able to run application pods on a cluster with an enabled PodSecurityPolicy admission controller unless these prerequisites are met. We strongly recommend you use PodSecurityPolicy admission controllers as follows:

- After you have created a new cluster, and tested your pod security polices, enable the Pod Security Admission Controller.
- Immediately after creating a new cluster, create roles (or clusterroles) and rolebindings (or clusterrolebindings).

References:

1. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies

CIS Controls:

Version 6

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.

Version 7

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.

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 PodSecurityPolicy (PSP) 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 PSP and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that PSP.

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

```
kubectl get psp <name> -o=jsonpath='{.spec.hostPID}'
```

Verify that there is at least one PSP which does not return true.

Remediation:

Create a PSP as described in the Kubernetes documentation, ensuring that the `.spec.hostPID` field is omitted or set to the correct value.

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

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.

Version 7

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.

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 PodSecurityPolicy (PSP) defined which does not permit containers to share the host IPC namespace.

If you have a requirement to containers which require hostIPC, this should be defined in a separate PSP and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that PSP.

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

```
kubectl get psp <name> -o=jsonpath='{.spec.hostIPC}'
```

Verify that there is at least one PSP which does not return true.

Remediation:

Create a PSP as described in the Kubernetes documentation, ensuring that the `.spec.hostIPC` field is omitted or set to the correct value.

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

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.

Version 7

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.

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 PodSecurityPolicy (PSP) defined which does not permit containers to share the host network namespace.

If you have need to run containers which require hostNetwork, this should be defined in a separate PSP and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that PSP.

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

```
kubectl get psp <name> -o=jsonpath='{.spec.hostNetwork}'
```

Verify that there is at least one PSP which does not return true.

Remediation:

Create a PSP as described in the Kubernetes documentation, ensuring that the `.spec.hostNetwork` field is omitted or set to the correct value.

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

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.

Version 7

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.

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.

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 PodSecurityPolicy (PSP) 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 PSP and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that PSP.

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

```
kubectl get psp <name> -o=jsonpath='{.spec.allowPrivilegeEscalation}'
```

Verify that there is at least one PSP which does not return true.

Remediation:

Create a PSP as described in the Kubernetes documentation, ensuring that the `.spec.allowPrivilegeEscalation` field is omitted or set to the correct value.

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

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.

Version 7

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.

4.2.6 Minimize the admission of root containers (Automated)

Profile Applicability:

- Level 2

Description:

Do not generally permit containers to be run as the root user.

Rationale:

Containers may run as any Linux user. Containers which run as the root user, whilst constrained by Container Runtime security features still have a escalated likelihood of container breakout.

Ideally, all containers should run as a defined non-UID 0 user.

There should be at least one PodSecurityPolicy (PSP) defined which does not permit root users in a container.

If you need to run root containers, this should be defined in a separate PSP and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that PSP.

Impact:

Pods with containers which run as the root user will not be permitted.

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether running containers as root is enabled:

```
kubectl get psp <name> -o=jsonpath='{.spec.runAsUser.rule}'
```

Verify that there is at least one PSP which returns `MustRunAsNonRoot` or `MustRunAs` with the range of UIDs not including 0.

Remediation:

Create a PSP as described in the Kubernetes documentation, ensuring that the `.spec.runAsUser.rule` is set to either `MustRunAsNonRoot` or `MustRunAs` with the range of UIDs not including 0.

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

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.

Version 7

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.

4.2.7 Minimize the admission of containers with the NET_RAW capability (Manual)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers with the potentially dangerous NET_RAW capability.

Rationale:

Containers run with a default set of capabilities as assigned by the Container Runtime. By default this can include potentially dangerous capabilities. With Docker as the container runtime the NET_RAW capability is enabled which may be misused by malicious containers.

Ideally, all containers should drop this capability.

There should be at least one PodSecurityPolicy (PSP) defined which prevents containers with the NET_RAW capability from launching.

If you need to run containers with this capability, this should be defined in a separate PSP and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that PSP.

Impact:

Pods with containers which run with the NET_RAW capability will not be permitted.

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether NET_RAW is disabled:

```
kubectl get psp <name> -o=jsonpath='{.spec.requiredDropCapabilities}'
```

Verify that there is at least one PSP which returns NET_RAW or ALL.

Remediation:

Create a PSP as described in the Kubernetes documentation, ensuring that the `.spec.requiredDropCapabilities` is set to include either `NET_RAW` or `ALL`.

Default Value:

By default, PodSecurityPolicies are not defined.

References:

1. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/
3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Version 6

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.

Version 7

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.

4.2.8 Minimize the admission of containers with added capabilities (Automated)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers with capabilities assigned beyond the default set.

Rationale:

Containers run with a default set of capabilities as assigned by the Container Runtime. Capabilities outside this set can be added to containers which could expose them to risks of container breakout attacks.

There should be at least one PodSecurityPolicy (PSP) defined which prevents containers with capabilities beyond the default set from launching.

If you need to run containers with additional capabilities, this should be defined in a separate PSP and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that PSP.

Impact:

Pods with containers which require capabilities outwith the default set will not be permitted.

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

Verify that there are no PSPs present which have `allowedCapabilities` set to anything other than an empty array.

Remediation:

Ensure that `allowedCapabilities` is not present in PSPs for the cluster unless it is set to an empty array.

Default Value:

By default, PodSecurityPolicies are not defined.

References:

1. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/
3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Version 6

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.

Version 7

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.

4.2.9 Minimize the admission of containers with capabilities assigned (Manual)

Profile Applicability:

- Level 2

Description:

Do not generally permit containers with capabilities

Rationale:

Containers run with a default set of capabilities as assigned by the Container Runtime. Capabilities are parts of the rights generally granted on a Linux system to the root user.

In many cases applications running in containers do not require any capabilities to operate, so from the perspective of the principal of least privilege use of capabilities should be minimized.

Impact:

Pods with containers require capabilities to operate will not be permitted.

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether capabilities have been forbidden:

```
kubectl get psp <name> -o=jsonpath='{.spec.requiredDropCapabilities}'
```

Remediation:

Review the use of capabilites in applications runnning on your cluster. Where a namespace contains applicaions which do not require any Linux capabities to operate consider adding a PSP which forbids the admission of containers which do not drop all capabilities.

Default Value:

By default, PodSecurityPolicies are not defined.

References:

1. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/
3. https://docs.cloud.oracle.com/en-us/iaas/Content/ContEng/Concepts/contengoverview.htm

CIS Controls:

Version 6

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.

Version 7

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.

4.3 CNI Plugin

4.3.1 Ensure latest CNI version is used (Manual)

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.

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:

Version 7

18.4 Only Use Up-to-date And Trusted Third-Party Components
Only use up-to-date and trusted third-party components for the software developed by the organization.

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.

```
kubectl get networkpolicy --all-namespaces

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:

Version 6

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.

Version 7

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).

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.

4.4 Secrets Management

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

Profile Applicability:

- Level 2

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
```

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:

Version 7

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

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.

4.4.2 Consider external secret storage (Manual)

Profile Applicability:

- Level 2

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:

Version 7

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.

4.5 Extensible Admission Control

4.6 General Policies

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

4.6.1 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:

Version 6

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

Version 7

14 Controlled Access Based on the Need to Know
Controlled Access Based on the Need to Know

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

Profile Applicability:

- Level 2

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.

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:

Version 6

3 Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers
Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers

Version 7

5 Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers
Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers

4.6.3 The default namespace should not be used (Manual)

Profile Applicability:

- Level 2

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

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:

Version 7

5.1 Establish Secure Configurations

Maintain documented, standard security configuration standards for all authorized operating systems and software.

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:

Version 7

3 Continuous Vulnerability Management
Continuous Vulnerability Management

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.

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:

Version 7

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.

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:

Version 7

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.

5.1.4 Minimize Container Registries to only those approved (Manual)

Profile Applicability:

- Level 1

Description:

Use approved container registries.

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:

Remediation:

CIS Controls:

Version 7

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.

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.

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 (Manual)

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.

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:

Version 7

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:

Version 7

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.

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 (Manual)

Profile Applicability:

- Level 1

Description:

Enable Master Authorized Networks to restrict access to the cluster's control plane (master endpoint) to only an allowlist (whitelist) of authorized IPs.

Rationale:

Authorized networks are a way of specifying a restricted range of IP addresses that are permitted to access your cluster's control plane. Kubernetes Engine uses both Transport Layer Security (TLS) and authentication to provide secure access to your cluster's control plane from the public internet. This provides you the flexibility to administer your cluster from anywhere; however, you might want to further restrict access to a set of IP addresses that you control. You can set this restriction by specifying an authorized network.

Restricting access to an authorized network can provide additional security benefits for your container cluster, including:

- Better protection from outsider attacks: Authorized networks provide an additional layer of security by limiting external, non-OCP access to a specific set of addresses you designate, such as those that originate from your premises. This helps protect access to your cluster in the case of a vulnerability in the cluster's authentication or authorization mechanism.
- Better protection from insider attacks: Authorized networks help protect your cluster from accidental leaks of master certificates from your company's premises. Leaked certificates used from outside OCP and outside the authorized IP ranges (for example, from addresses outside your company) are still denied access.

Impact:

When implementing Master Authorized Networks, be careful to ensure all desired networks are on the allowlist (whitelist) to prevent inadvertently blocking external access to your cluster's control plane.

Audit:

Remediation:

Default Value:

By default, Master Authorized Networks is disabled.

CIS Controls:

Version 7

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.

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

Profile Applicability:

- Level 1

Description:

Disable access to the Kubernetes API from outside the node network if it is not required.

Rationale:

In a private cluster, the master node has two endpoints, a private and public endpoint. The private endpoint is the internal IP address of the master, behind an internal load balancer in the master's VPC network. Nodes communicate with the master using the private endpoint. The public endpoint enables the Kubernetes API to be accessed from outside the master's VPC network.

Although Kubernetes API requires an authorized token to perform sensitive actions, a vulnerability could potentially expose the Kubernetes publically with unrestricted access. Additionally, an attacker may be able to identify the current cluster and Kubernetes API version and determine whether it is vulnerable to an attack. Unless required, disabling public endpoint will help prevent such threats, and require the attacker to be on the master's VPC network to perform any attack on the Kubernetes API.

Impact:

This topic gives an overview of the options for enabling private access to services within Oracle Cloud Infrastructure. Private access means that traffic does not go over the internet. Access can be from hosts within your virtual cloud network (VCN) or your on-premises network.

- You can enable private access to certain services within Oracle Cloud Infrastructure from your VCN or on-premises network by using either a private endpoint or a service gateway. See the sections that follow.
- For each private access option, these services or resource types are available:
  - With a private endpoint: Autonomous Database (shared Exadata infrastructure)
  - With a service gateway: Available services
- With either private access option, the traffic stays within the Oracle Cloud Infrastructure network and does not traverse the internet. However, if you use a service gateway, requests to the service use a public endpoint for the service.

- If you do not want to access a given Oracle service through a public endpoint, Oracle recommends using a private endpoint in your VCN (assuming the service supports private endpoints). A private endpoint is represented as a private IP address within a subnet in your VCN.

See About Private Endpoints

Audit:

Remediation:

Default Value:

By default, the Private Endpoint is disabled.

CIS Controls:

Version 7

12 Boundary Defense
Boundary Defense

5.4.3 Ensure clusters are created with Private Nodes (Manual)

Profile Applicability:

- Level 1

Description:

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:

Remediation:

Default Value:

By default, Private Nodes are disabled.

CIS Controls:

Version 7

12 Boundary Defense
Boundary Defense

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

Profile Applicability:

- Level 1

Description:

Use 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:

Remediation:

Default Value:

By default, Network Policy is disabled.

CIS Controls:

Version 7

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.

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.

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:

Remediation:

CIS Controls:

Version 7

14.4 Encrypt All Sensitive Information in Transit
Encrypt all sensitive information in transit.

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:

Version 7

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

Control		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)	☐	☐
2.1.2	Ensure OKE service level admins are created to manage OKE (Manual)	☐	☐
2.2	**Authentication and Authorization**		
2.2.1	Client certificate authentication should not be used for users (Manual)	☐	☐
2.2.2	Ensure OKE service level admins are created to manage OKE (Manual)	☐	☐
2.3	**Logging**		
2.3.1	Ensure access to OCI Audit service Log for OKE (Automated)	☐	☐
2.3.2	Ensure that the audit policy covers key security concerns (Manual)	☐	☐
3	**Worker Nodes**		
3.1	**Worker Node Configuration Files**		
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive (Manual)	☐	☐
3.1.2	Ensure that the proxy kubeconfig file ownership is set to root:root (Manual)	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Manual)	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root (Manual)	☐	☐
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)	☐	☐
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 (Manual)	☐	☐

3.2.6	Ensure that the --protect-kernel-defaults argument is set to true (Manual)	☐	☐
3.2.7	Ensure that the --make-iptables-util-chains argument is set to true (Automated)	☐	☐
3.2.8	Ensure that the --hostname-override argument is not set (Manual)	☐	☐
3.2.9	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Automated)	☐	☐
3.2.10	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	☐	☐
3.2.11	Ensure that the --rotate-certificates argument is not set to false (Automated)	☐	☐
3.2.12	Ensure that the --rotate-server-certificates argument is set to true (Manual)	☐	☐
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. (Manual)	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary (Manual)	☐	☐
4.2	**Pod Security Policies**		
4.2.1	Minimize the admission of privileged containers (Manual)	☐	☐
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)	☐	☐
4.2.6	Minimize the admission of root containers (Automated)	☐	☐
4.2.7	Minimize the admission of containers with the NET_RAW capability (Manual)	☐	☐
4.2.8	Minimize the admission of containers with added capabilities (Automated)	☐	☐
4.2.9	Minimize the admission of containers with capabilities assigned (Manual)	☐	☐
4.3	**CNI Plugin**		
4.3.1	Ensure latest CNI version is used (Manual)	☐	☐

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 (Manual)	☐	☐
4.4.2	Consider external secret storage (Manual)	☐	☐
4.5	**Extensible Admission Control**		
4.6	**General Policies**		
4.6.1	Create administrative boundaries between resources using namespaces (Manual)	☐	☐
4.6.2	Apply Security Context to Your Pods and Containers (Manual)	☐	☐
4.6.3	The default namespace should not be used (Manual)	☐	☐
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 (Manual)	☐	☐
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 (Manual)	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Manual)	☐	☐
5.4.3	Ensure clusters are created with Private Nodes (Manual)	☐	☐
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)	☐	☐
5.5	**Authentication and Authorization**		
5.5.1	Access Control and Container Engine for Kubernetes (Manual)	☐	☐