CIS Benchmarks

CIS RedHat OpenShift Container Platform Benchmark

v1.0.0 - 02-25-2021

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 establishing a secure configuration posture for OpenShift 4.5 and 4.6

**Note: **The set of configuration files mentioned throughout this benchmark are specific to Red Hat's CNCF certified Kubernetes distribution, Red Hat OpenShift Container Platform. Each section includes information about the default configuration of an OpenShift cluster and a set of recommendations for hardening the configuration, inspired by the CIS Kubernetes benchmark. For each hardening recommendation, information on how to implement the control and/or how to verify or audit the control is provided. In some cases, remediation information is also provided.

The majority of the settings in the hardening guide are in place by default. The audit information for these settings is provided so that you can verify that the cluster admin has not made changes that would be less secure than the OpenShift defaults. A small number of items require configuration. Finally, there are some recommendations that require decisions by the customer, such as audit log size, retention and related settings.

The recommendations that require decisions based on your needs are:

- Configure encryption of data at rest in etcd datastore
- Manage Image Provenance
- Set the --event-qps argument as appropriate
- Configure the API server audit log retention
- Configure cluster logging to forward audit logs off the cluster
- Configure the audit log file size
- Adjust the garbage collection settings as needed
- Create custom Security Context Constraints as needed
- Configure Network Policies as appropriate
- In OCP 4.6 and above, configure audit policies as appropriate

Intended Audience

This document is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel who plan to develop, deploy, assess, or secure solutions that incorporate OpenShift 4.5 or 4.6.

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**

  Items in this profile intend to:

  - be practical and prudent;
  - provide a clear security benefit; and
  - not inhibit the utility of the technology beyond acceptable means.

- **Level 2**

  This profile extends the "Level 1" profile. Items in this profile exhibit one or more of the following characteristics:

  - are intended for environments or use cases where security is paramount
  - acts as defense in depth measure
  - may negatively inhibit the utility or performance of the technology.

Acknowledgements

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

Author/s
Randall Mowen
Kirsten Newcomer

Contributor/s
RedHat Openshift Team
Mark Larinde

Recommendations

1 Control Plane Components

This section consists of security recommendations for the direct configuration of Kubernetes control plane processes. These recommendations assume that the OpenShift cluster has 3 master nodes, as that is the default configuration on installation. These recommendations may not be directly applicable for cluster operators in environments where these components are managed by a 3rd party such as OpenShift Dedicated, Azure Red Hat OpenShift or Red Hat OpenShift Service on AWS.

All Audit and Remediation commands assume that you are logged into the OpenShift cluster with the cluster admin role, cluster bound.

1.1 Master Node Configuration Files

1.1.1 Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the API server pod specification file has permissions of `644` or more restrictive.

Rationale:

The API server pod specification file controls various parameters that set the behavior of the API server. You should restrict its file permissions to maintain the integrity of the file. The file should be writable only by the administrators on the system.

Impact:

None

Audit:

OpenShift 4 deploys two API servers: the OpenShift API server and the Kube API server. The OpenShift API server delegates requests for Kubernetes objects to the Kube API server. The OpenShift API server is managed as a deployment. The pod specification yaml for openshift-apiserver is stored in etcd.
The Kube API Server is managed as a static pod. The pod specification file for the kube-apiserver is created on the control plane nodes at /etc/kubernetes/manifests/kube-apiserver-pod.yaml. The kube-apiserver is mounted via hostpath to the kube-apiserver pods via /etc/kubernetes/static-pod-resources/kube-apiserver-pod.yaml with permissions 0644.
To verify pod specification file permissions for the kube-apiserver, run the following command.

```
#echo "check kube-apiserver pod specification file permissions"

for i in $( oc get pods -n openshift-kube-apiserver -l app=openshift-kube-
apiserver -o name )
do
  oc exec  -n openshift-kube-apiserver $i -- \
```

```
  stat -c %a  /etc/kubernetes/static-pod-resources/kube-apiserver-pod.yaml
done
```

Verify that the permissions are 644 or more restrictive.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, the kube-apiserver-pod.yaml file has permissions of 644.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

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.

1.1.2 Ensure that the API server pod specification file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the API server pod specification file ownership is set to `root:root`.

Rationale:

The API server pod specification file controls various parameters that set the behavior of the API server. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Impact:

None

Audit:

OpenShift 4 deploys two API servers: the OpenShift API server and the Kube API server. The OpenShift API server is managed as a deployment. The pod specification yaml for openshift-apiserver is stored in etcd.
The Kube API Server is managed as a static pod. The pod specification file for the kube-apiserver is created on the control plane nodes at /etc/kubernetes/manifests/kube-apiserver-pod.yaml. The kube-apiserver is mounted via hostpath to the kube-apiserver pods via /etc/kubernetes/static-pod-resources/kube-apiserver-pod.yaml with ownership `root:root`.
To verify pod specification file ownership for the kube-apiserver, run the following command.

```
#echo "check kube-apiserver pod specification file ownership"

for i in $( oc get pods -n openshift-kube-apiserver -l app=openshift-kube-
apiserver -o name )
do
  oc exec  -n openshift-kube-apiserver $i -- \
  stat -c %U:%G  /etc/kubernetes/static-pod-resources/kube-apiserver-pod.yaml
done
```

Verify that the ownership is set to `root:root`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, the kube-apiserver-pod.yaml file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

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.

1.1.3 Ensure that the controller manager pod specification file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the controller manager pod specification file has permissions of `644` or more restrictive.

Rationale:

The controller manager pod specification file controls various parameters that set the behavior of the Controller Manager on the master 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.

Impact:

None

Audit:

OpenShift 4 deploys two controller managers: the OpenShift Controller manager and the Kube Controller manager.
The OpenShift Controller manager is managed as a deployment. The pod specification yaml for openshift-controller-manager is stored in etcd.
The Kube Controller manager is managed as a static pod. The pod specification file for the openshift-kube-controller-manager is created on control plane nodes at /etc/kubernetes/manifests/kube-controller-manager-pod.yaml. It is mounted via hostpath to the kube-controller-manager pods via /etc/kubernetes/static-pod-resources/kube-controller-manager-pod.yaml with permissions 0644.
To verify pod specification file permissions for the kube-controller-manager, run the following command.

```
#echo "check openshift-kube-controller-manager pod specification file
permissions"

for i in $( oc get pods -n openshift-kube-controller-manager -o name -l
app=kube-controller-manager)
do
  oc exec  -n openshift-kube-controller-manager $i -- \
  stat -c %a /etc/kubernetes/static-pod-resources/kube-controller-manager-
```

```
pod.yaml
done
```

Verify that the permissions are 644 or more restrictive.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, the `kube-controller-manager-pod.yaml` file has permissions of `644`.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/

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.

1.1.4 Ensure that the controller manager pod specification file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the controller manager pod specification file ownership is set to `root:root`.

Rationale:

The controller manager pod specification file controls various parameters that set the behavior of various components of the master 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:

OpenShift 4 deploys two controller managers: the OpenShift Controller manager and the Kube Controller manager.
The OpenShift Controller manager is managed as a deployment. The pod specification yaml for openshift-controller-manager is stored in etcd.
The Kube Controller manager is managed as a static pod. The pod specification file for the openshift-kube-controller-manager is created on control plane nodes at /etc/kubernetes/manifests/kube-controller-manager-pod.yaml. It is mounted via hostpath to the kube-controller-manager pods via /etc/kubernetes/static-pod-resources/kube-controller-manager-pod.yaml with ownership root:root.
Run the following command.

```
#echo "openshift-kube-controller-manager pod specification file ownership"

for i in $( oc get pods -n openshift-kube-controller-manager -o name -l
app=kube-controller-manager)
do
  oc exec  -n openshift-kube-controller-manager $i -- \
  stat -c %U:%G /etc/kubernetes/static-pod-resources/kube-controller-manager-
pod.yaml
done
```

Verify that the ownership is set to `root:root`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, the `kube-controller-manager-pod.yaml` file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/

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.

1.1.5 Ensure that the scheduler pod specification file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the scheduler pod specification file has permissions of `644` or more restrictive.

Rationale:

The scheduler pod specification file controls various parameters that set the behavior of the Scheduler service in the master 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.

Impact:

None

Audit:

In OpenShift 4 the kube-scheduler is deployed as a static pod and its pod specification file is created on control plane nodes at /etc/kubernetes/manifests/kube-scheduler-pod.yaml. It is mounted via hostpath to the kube-controller-manager pods via /etc/kubernetes/static-pod-resources/kube-scheduler-pod.yaml with permissions 0644.
To verify, run the following command.

```
#Verify openshift-kube-scheduler permissions

for i in $(oc get pods -n openshift-kube-scheduler -l app=openshift-kube-
scheduler -o name)
 do
  oc exec -n openshift-kube-scheduler $i -- \
  stat -c %a /etc/kubernetes/static-pod-resources/kube-scheduler-pod.yaml
 done
```

Verify that the permissions are `644` or more restrictive.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, `openshift-kube-scheduler-pod.yaml` file has permissions of `644`.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-kube-scheduler-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.3/nodes/scheduling/nodes-scheduler-about.html
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/

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.

1.1.6 Ensure that the scheduler pod specification file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the scheduler pod specification file ownership is set to `root:root`.

Rationale:

The scheduler pod specification file controls various parameters that set the behavior of the `kube-scheduler` service in the master 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:

In OpenShift 4, the kube-scheduler is deployed as a static pod and its pod specification file is created on control plane nodes at /etc/kubernetes/manifests/kube-scheduler-pod.yaml. It is mounted via hostpath to the kube-controller-manager pods via /etc/kubernetes/static-pod-resources/kube-scheduler-pod.yaml with ownership `root:root`.
Run the following command.

```
#Verify openshift-kube-scheduler ownership
for i in $(oc get pods -n openshift-kube-scheduler -l app=openshift-kube-
scheduler -o name)
 do
  oc exec -n openshift-kube-scheduler $i -- \
  stat -c %U:%G /etc/kubernetes/static-pod-resources/kube-scheduler-pod.yaml
 done
```

Verify that the ownership is set to `root:root`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, `kube-scheduler-pod.yaml` file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-kube-scheduler-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.3/nodes/scheduling/nodes-scheduler-about.html
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/

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.

1.1.7 Ensure that the etcd pod specification file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `/etc/kubernetes/manifests/etcd.yaml` file has permissions of `644` or more restrictive.

Rationale:

The etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` controls various parameters that set the behavior of the `etcd` service in the master node. `etcd` is a highly-available key-value store which Kubernetes uses for persistent storage of all of its REST API object. 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:

In OpenShift 4, starting with OpenShift 4.4, the etcd pod specification file is generated by the cluster etcd operator. The pod specification file is created on control plane nodes at `/etc/kubernetes/manifests/etcd-member.yaml` with permissions `644`.
The default etcd pod specification file is available here: [openshift/cluster-etcd-operator]
Run the following command.

```
#Verify openshift-etcd permissions

for i in $(oc get pods -n openshift-etcd -l app=etcd -o name | grep etcd )
do
  echo "check pod $i"
  oc rsh  -n openshift-etcd  $i \
        stat -c %a /etc/kubernetes/manifests/etcd-pod.yaml
done
```

Verify that the permissions are 644 or more restrictive.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, `/etc/kubernetes/manifests/etcd-pod.yaml` file has permissions of `644`.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://github.com/openshift/cluster-etcd-operator/blob/master/bindata/etcd/pod.yaml
3. https://etcd.io/
4. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/

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.

1.1.8 Ensure that the etcd pod specification file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `/etc/kubernetes/manifests/etcd.yaml` file ownership is set to `root:root`.

Rationale:

The etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` controls various parameters that set the behavior of the `etcd` service in the master node. `etcd` is a highly-available key-value store which Kubernetes uses for persistent storage of all of its REST API object. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Impact:

None

Audit:

In OpenShift 4, starting with OpenShift 4.4, the etcd pod specification file is generated by the cluster etcd operator. The pod specification file is created on control plane nodes at `/etc/kubernetes/manifests/etcd-member.yaml` with ownership `root:root`.
The default etcd pod specification file is available here: openshift/cluster-etcd-operator
Run the following command :

```
#Verify openshift-etcd ownership
for i in $(oc get pods -n openshift-etcd -l app=etcd -o name | grep etcd )
do
  echo "check pod $i"
  oc rsh  -n openshift-etcd  $i \
        stat -c %U:%G /etc/kubernetes/manifests/etcd-pod.yaml
done
```

Verify that the ownership is set to `root:root`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, `/etc/kubernetes/manifests/etcd-member.yaml` file ownership is set to `root:root`.

References:

1. https://coreos.com/etcd
2. https://kubernetes.io/docs/admin/etcd/
3. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane

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.

1.1.9 Ensure that the Container Network Interface file permissions are set to 644 or more restrictive (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that the Container Network Interface files have permissions of `644` or more restrictive.

Rationale:

Container Network Interface provides various networking options for overlay networking. You should consult their documentation and restrict their respective file permissions to maintain the integrity of those files. Those files should be writable by only the administrators on the system.

Impact:

None

Audit:

The Cluster Network Operator (CNO) deploys and manages the cluster network components on an OpenShift Container Platform cluster. This includes the Container Network Interface (CNI) default network provider plug-in selected for the cluster during installation. OpenShift Container Platform uses the Multus CNI plug-in to allow chaining of CNI plug-ins. The default Pod network must be configured during cluster installation. By default, the CNO deploys the OpenShift SDN as the default Pod network.
Ensure that the Container Network Interface file permissions, multus, openshift-sdn and Open vSwitch (OVS) file permissions are set to 644 or more restrictive. The SDN components are deployed as DaemonSets across the master/worker nodes with controllers limited to control plane nodes. OpenShift deploys OVS as a network overlay by default.
Various configurations (ConfigMaps and other files managed by the operator via hostpath but stored on the container hosts) are stored in the following locations:
CNI/Multus (pod muluts):

```
/host/etc/cni/net.d = CNI_CONF_DIR
/host/var/run/multus/cni/net.d = multus config dir
```

SDN (pod ovs; daemonset; app=ovs):

```
/var/lib/cni/networks/openshift-sdn
/var/run/openshift-sdn
```

OVS (container openvswitch):

```
/var/run/openvswitch
/etc/openvswitch
/run/openvswitch
```

Run the following commands.

```
# needs verification

# For CNI multus
for i in $(oc get pods -n openshift-multus -l app=multus -oname); do  oc exec
-n openshift-multus $i --  /bin/bash -c "stat -c \"%a %n\"
/host/etc/cni/net.d/*.conf"; done

for i in $(oc get pods -n openshift-multus -l app=multus -oname); do  oc exec
-n openshift-multus $i --  /bin/bash -c "stat -c \"%a %n\"
/host/var/run/multus/cni/net.d/*.conf"; done

# For SDN pods
for i in $(oc get pods -n openshift-sdn -l app=sdn -oname);  do   oc exec -n
openshift-sdn $i --   find /var/lib/cni/networks/openshift-sdn -type f -exec
stat -c %a {} \;;  done

for i in $(oc get pods -n openshift-sdn -l app=sdn -oname); do  oc exec -n
openshift-sdn $i -- find /var/run/openshift-sdn -type f -exec stat -c %a {}
\;; done

# For OVS pods
for i in $(oc get pods -n openshift-sdn -l app=ovs -oname);  do   oc exec -n
openshift-sdn $i --   find /var/run/openvswitch -type f -exec stat -c %a {}
\;;  done

for i in $(oc get pods -n openshift-sdn -l app=ovs -oname);  do   oc exec -n
openshift-sdn $i --   find /etc/openvswitch -type f -exec stat -c %a {} \;;
done

for i in $(oc get pods -n openshift-sdn -l app=ovs -oname);  do   oc exec -n
openshift-sdn $i --   find /run/openvswitch -type f -exec stat -c %a {} \;;
done
```

Verify that the config files for the CNI multus pods have permissions of 644 or more restrictive.

```
/host/etc/cni/net.d/00-multus.conf = 600
/host/var/run/multus/cni/net.d/80-openshift-network.conf = 644
```

Verify that the SDN pods permissions are 644 or more restrictive.

```
/var/lib/cni/networks/openshift-sdn/* = 644
/var/run/openshift-sdn/cniserver/config.json = 444
```

Verify that the OVS permissions are 644 or more restrictive.

```
/var/run/openvswitch/ovs-vswitchd.pid = 644
/etc/openvswitch/conf.db = 644
/etc/openvswitch/system-id.conf = 644
/etc/openvswitch/.conf.db.~lock~ = 600
```

```
/run/openvswitch/ovs-vswitchd.pid = 644
/run/openvswitch/ovsdb-server.pid = 644
```

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

In OpenShift 4, the default values are:

```
/host/etc/cni/net.d/00-multus.conf = 600 /host/var/run/multus/cni/net.d/80-
openshift-network.conf = 644 /var/lib/cni/networks/openshift-sdn/* = 644
/var/run/openshift-sdn/cniserver/config.json = 444 /var/run/openvswitch/ovs-
vswitchd.pid = 644 /etc/openvswitch/conf.db = 644 /etc/openvswitch/system-
id.conf = 644 /etc/openvswitch/.conf.db.~lock~ = 600 /run/openvswitch/ovs-
vswitchd.pid = 644 /run/openvswitch/ovsdb-server.pid = 644
```

References:

1. https://docs.openshift.com/container-platform/4.3/networking/cluster-network-operator.html
2. https://kubernetes.io/docs/concepts/cluster-administration/networking/

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.

1.1.10 Ensure that the Container Network Interface file ownership is set to root:root (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that the Container Network Interface files have ownership set to `root:root`.

Rationale:

Container Network Interface provides various networking options for overlay networking. You should consult their documentation and restrict their respective file permissions to maintain the integrity of those files. Those files should be owned by `root:root`.

Impact:

None

Audit:

The Cluster Network Operator (CNO) deploys and manages the cluster network components on an OpenShift Container Platform cluster. This includes the Container Network Interface (CNI) default network provider plug-in selected for the cluster during installation. OpenShift Container Platform uses the Multus CNI plug-in to allow chaining of CNI plug-ins. The default Pod network must be configured during cluster installation. By default, the CNO deploys the OpenShift SDN as the default Pod network.
Ensure that the `multu` and `openshift-sdn` file ownership is set to root:root and the Open vSwitch (OVS) file ownership is set to `openvswitch:openvswitch`.
The SDN components are deployed as DaemonSets across the master/worker nodes with controllers limited to control plane nodes. OpenShift deploys OVS as a network overlay by default. Various configurations (ConfigMaps and other files managed by the operator via `hostpath` but stored on the container hosts) are stored in the following locations:
CNI:
```
/etc/cni/net.d
/host/var/run/multus/cni/net.d
```
SDN:
```
/var/lib/cni/networks/openshift-sdn
/var/run/openshift-sdn
```
SDN OVS:
```
/var/run/openvswitch
```

/etc/openvswitch
/run/openvswitch

Run the following commands.

```
# needs verification

# For CNI multus
for i in $(oc get pods -n openshift-multus -l app=multus -oname); do  oc exec
-n openshift-multus $i --  /bin/bash -c "stat -c \"%U:%G %n\"
/host/etc/cni/net.d/*.conf"; done

for i in $(oc get pods -n openshift-multus -l app=multus -oname); do  oc exec
-n openshift-multus $i --  /bin/bash -c "stat -c \"%U:%G %n\"
/host/var/run/multus/cni/net.d/*.conf"; done

# For SDN pods
for i in $(oc get pods -n openshift-sdn -l app=sdn -oname);  do   oc exec -n
openshift-sdn $i --   find /var/lib/cni/networks/openshift-sdn -type f -exec
stat -c \"%U:%G\" {} \;;  done

for i in $(oc get pods -n openshift-sdn -l app=sdn -oname); do  oc exec -n
openshift-sdn $i -- find /var/run/openshift-sdn -type f -exec stat -c %U:%G
{} \;; done

# For OVS pods in 4.5
for i in $(oc get pods -n openshift-sdn -l app=ovs -oname);  do   oc exec -n
openshift-sdn $i --   find /var/run/openvswitch -type f -exec stat -c %U:%G
{} \;;  done

for i in $(oc get pods -n openshift-sdn -l app=ovs -oname);  do   oc exec -n
openshift-sdn $i --   find /etc/openvswitch -type f -exec stat -c %U:%G {}
\;;  done

for i in $(oc get pods -n openshift-sdn -l app=ovs -oname);  do   oc exec -n
openshift-sdn $i --   find /run/openvswitch -type f -exec stat -c %U:%G {}
\;;  done

# For OVS pods in 4.6
TBD
```

Verify that the CNI and SDN file ownership is set to `root:root`.

```
/host/etc/cni/net.d/00-multus.conf = root:root
/host/var/run/multus/cni/net.d/80-openshift-network.conf = root:root
/var/lib/cni/networks/openshift-sdn = root:root
/var/run/openshift-sdn = root:root
```

Verify that the OVS file ownership is set to `openvswitch:openvswitch`.

```
/var/run/openvswitch = openvswitch:openvswitch
/etc/openvswitch = openvswitch:openvswitch
/run/openvswitch = openvswitch:openvswitch
```

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

In OpenShift 4, the default file ownership is root:root for CNI Multus and SDN and openvswitch:openvswitch for the OVS plugin.

```
/host/etc/cni/net.d/00-multus.conf = root:root
/host/var/run/multus/cni/net.d/80-openshift-network.conf = root:root
/var/lib/cni/networks/openshift-sdn = root:root /var/run/openshift-sdn =
root:root /var/run/openvswitch = openvswitch:openvswitch /etc/openvswitch =
openvswitch:openvswitch /run/openvswitch = openvswitch:openvswitch
```

References:

1. https://docs.openshift.com/container-platform/4.3/networking/cluster-network-operator.html
2. https://kubernetes.io/docs/concepts/cluster-administration/networking/

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.

1.1.11 Ensure that the etcd data directory permissions are set to 700 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the etcd data directory has permissions of `700` or more restrictive.

Rationale:

`etcd` is a highly-available key-value store used by Kubernetes deployments for persistent storage of all of its REST API objects. This data directory should be protected from any unauthorized reads or writes. It should not be readable or writable by any group members or the world.

Impact:

None

Audit:

In OpenShift 4, `etcd` members are deployed on the master nodes as static pods. The pod specification file is created on control plane nodes at `/etc/kubernetes/manifests/etcd-member.yaml`. The `etcd` database is stored on the container host in `/var/lib/etcd` and mounted to the `etcd-member` container via the host path mount data-dir with the same filesystem path (`/var/lib/etcd`). The permissions for this directory on the container host is `700`.
Starting with OCP 4.4, `etcd` is managed by the `cluster-etcd-operator`. The `etcd` operator will help to automate restoration of master nodes. There is also a new `etcdctl` container in the `etcd` static pod for quick debugging. cluster-admin rights are required to exec into `etcd` containers.
Run the following commands.

```
for i in $(oc get pods -n openshift-etcd -l app=etcd -oname); do oc exec -n
openshift-etcd -c etcd $i -- stat -c %a%n /var/lib/etcd/member; done
```

Verify that the permissions are `700`.

Remediation:

No remediation required. File permissions are managed by the `etcd` operator.

Default Value:

By default, `etcd` data directory has permissions of `700`.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://etcd.io/#data-dir
3. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/

CIS Controls:

Version 6

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

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.

1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `etcd` data directory ownership is set to `root:root`.

Rationale:

`etcd` is a highly-available key-value store used by Kubernetes deployments for persistent storage of all of its REST API objects. This data directory should be protected from any unauthorized reads or writes. It should be owned by `root:root`.

NOTE: The only users that exist on an RHCOS OpenShift node are root and core. This is intentional, as regular management of the underlying RHCOS cluster nodes is designed to be performed via the OpenShift API itself. The core user is a member of the wheel group, which gives it permission to use `sudo` for running privileged commands. Adding additional users at the node level is highly discouraged.

Impact:

None

Audit:

In OpenShift 4, `etcd` members are deployed on the master nodes as static pods. The `etcd` database is stored on the master nodes in `/var/lib/etcd` and mounted to the `etcd-member` container via the host path mount data-dir with the same filesystem path (`/var/lib/etcd`). The ownership for this directory on the `etcd-member` container and on the container host is `root:root`.
Starting with OCP 4.4, `etcd` is managed by the `cluster-etcd-operator`. The `etcd` operator will help to automate restoration of master nodes. There is also a new `etcdctl` container in the `etcd` static pod for quick debugging. cluster-admin rights are required to exec into `etcd` containers.
Run the following command.

```
for i in $(oc get pods -n openshift-etcd -l app=etcd -oname); do oc exec -n
openshift-etcd -c etcd $i -- stat -c %U:%G /var/lib/etcd/member; done
```

Verify that the ownership is set to `root:root`.

Remediation:

No remediation required; file ownership is managed by the operator.

Default Value:

By default, in OpenShift 4, `etcd` data directory ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.3/architecture/control-plane.html#defining-masters_control-plane
2. https://etcd.io/#data-dir
3. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/

CIS Controls:

Version 6

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

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.

1.1.13 Ensure that the admin.conf file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `admin.conf` file has permissions of `644` or more restrictive.

Rationale:

The `admin.conf` is the administrator `kubeconfig` file defining various settings for the administration of the cluster. 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:

In OpenShift 4 the admin config file is stored in `/etc/kubernetes/kubeconfig` with permissions `644`.
Run the following command.

```
for i in $(oc get nodes  -o name)
 do
  echo $i
  oc debug $i --  <<EOF
        chroot /host
        stat -c%a /etc/kubernetes/kubeconfig
EOF
 done
```

Verify that the permissions are `644` or more restrictive.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, `kubeconfig` has permissions of `644`.

References:

1. https://docs.openshift.com/container-platform/4.5/cli_reference/openshift_cli/administrator-cli-commands.html
2. https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

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.

1.1.14 Ensure that the admin.conf file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `admin.conf` file ownership is set to `root:root`.

Rationale:

The `admin.conf` file contains the admin credentials for the cluster. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Impact:

None.

Audit:

In OpenShift 4 the admin config file is stored in `/etc/kubernetes/kubeconfig` with ownership `root:root`.
Run the following command.

```
for i in $(oc get nodes  -o name)
 do
  echo $i
  oc debug $i --  <<EOF
        chroot /host
        stat -c %U:%G  /etc/kubernetes/kubeconfig
EOF
 done
```

Verify that the ownership is set to `root:root`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, in OpenShift 4, `kubeconfig` file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.5/cli_reference/openshift_cli/administrator-cli-commands.html
2. https://kubernetes.io/docs/reference/setup-tools/kubeadm/

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.

1.1.15 Ensure that the scheduler.conf file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `scheduler.conf` file has permissions of `644` or more restrictive.

Rationale:

The `scheduler.conf` file is the `kubeconfig` file for the Scheduler. 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:

The `kubeconfig` file for `kube-scheduler` is stored in the ConfigMap `scheduler-kubeconfig` in the namespace `openshift-kube-scheduler`. The file `kubeconfig` (`scheduler.conf`) is referenced in the pod via `hostpath` and is stored in `/etc/kubernetes/static-pod-resources/configmaps/scheduler-kubeconfig/kubeconfig` with permissions `644`.
Run the following command :

```
for i in $(oc get pods -n openshift-kube-scheduler -l app=openshift-kube-
scheduler -o name)
 do
  oc exec -n openshift-kube-scheduler $i -- \
  stat -c %a /etc/kubernetes/static-pod-resources/configmaps/scheduler-
kubeconfig/kubeconfig
 done
```

Verify that the permissions are `644` or more restrictive.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, `scheduler-kubeconfig/kubeconfig` has permissions of `644`.

References:

1. https://docs.openshift.com/container-platform/4.4/operators/operator-reference.html#cluster-kube-scheduler-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.3/nodes/scheduling/nodes-scheduler-about.html
3. https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/

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.

1.1.16 Ensure that the scheduler.conf file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `scheduler.conf` file ownership is set to `root:root`.

Rationale:

The `scheduler.conf` file is the kubeconfig file for the Scheduler. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Impact:

None

Audit:

The `kubeconfig` file for `kube-scheduler` is stored in the ConfigMap `scheduler-kubeconfig` in the namespace `openshift-kube-scheduler`. The file `kubeconfig` (scheduler.conf) is referenced in the pod via `hostpath` and is stored in `/etc/kubernetes/static-pod-resources/configmaps/scheduler-kubeconfig/kubeconfig` with ownership `root:root`. Run the following command.

```
for i in $(oc get pods -n openshift-kube-scheduler -l app=openshift-kube-
scheduler -o name)
 do
  oc exec -n openshift-kube-scheduler $i -- \
  stat -c %U:%G /etc/kubernetes/static-pod-resources/configmaps/scheduler-
kubeconfig/kubeconfig
 done
```

Verify that the ownership is set to `root:root`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, `scheduler-kubeconfig/kubeconfig` file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-kube-scheduler-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.3/nodes/scheduling/nodes-scheduler-about.html
3. https://kubernetes.io/docs/concepts/scheduling-eviction/kube-scheduler/

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.

1.1.17 Ensure that the controller-manager.conf file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `controller-manager.conf` file has permissions of `644` or more restrictive.

Rationale:

The `controller-manager.conf` file is the `kubeconfig` file for the Controller Manager. 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:

The `kubeconfig` file for `kube-controller-manager` is stored in the ConfigMap `controller-manager-kubeconfig` in the namespace `openshift-kube-controller-manager`. The file `kubeconfig` (`controller-manager.conf`) is referenced in the pod via `hostpath` and is stored in `/etc/kubernetes/static-pod-resources/configmaps/controller-manager-kubeconfig/kubeconfig` with permissions `644`.
Run the following command.

```
for i in $(oc get pods -n openshift-kube-controller-manager -l app=kube-
controller-manager -oname)
 do
  oc exec -n openshift-kube-controller-manager $i -- \
  stat -c %a /etc/kubernetes/static-pod-resources/configmaps/controller-
manager-kubeconfig/kubeconfig
 done
```

Verify that the permissions are `644` or more restrictive.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, the `controller-manager-kubeconfig/kubeconfig` file has permissions of `640`.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/

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.

1.1.18 Ensure that the controller-manager.conf file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `controller-manager.conf` file ownership is set to `root:root`.

Rationale:

The `controller-manager.conf` file is the `kubeconfig` file for the Controller Manager. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Impact:

None

Audit:

Run the following command:

```
for i in $(oc get pods -n openshift-kube-controller-manager -l app=kube-
controller-manager -oname)
 do
  oc exec -n openshift-kube-controller-manager $i -- \
  stat -c %U:%G /etc/kubernetes/static-pod-resources/configmaps/controller-
manager-kubeconfig/kubeconfig
 done
```

Verify that the ownership is set to `root:root`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, `controller-manager-kubeconfig` file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/

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.

1.1.19 Ensure that the OpenShift PKI directory and file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the OpenShift PKI directory and file ownership is set to `root:root`.

Rationale:

OpenShift makes use of a number of certificates as part of its operation. You should verify the ownership of the directory containing the PKI information and all files in that directory to maintain their integrity. The directory and files should be owned by `root:root`.

Impact:

None

Audit:

Keys for control plane components deployed as static pods, `kube-apiserver`, `kube-controller-manager`, and `openshift-kube-scheduler` are stored in the directory `/etc/kubernetes/static-pod-certs/secrets`. The directory and file ownership are set to `root:root`.
Run the following command.

```
# Should return root:root for all files and directories

for i in $(oc -n openshift-kube-apiserver get pod -l app=openshift-kube-
apiserver -o jsonpath='{.items[*].metadata.name}')
do
        echo $i static-pod-certs
        oc exec -n openshift-kube-apiserver $i -c kube-apiserver -- \
        find /etc/kubernetes/static-pod-certs -type d -wholename '*/secrets*'
-exec stat -c %U:%G {} \;
        oc exec -n openshift-kube-apiserver $i -c kube-apiserver -- \
        find /etc/kubernetes/static-pod-certs -type f -wholename '*/secrets*'
-exec stat -c %U:%G {} \;
        echo $i static-pod-resources
        oc exec -n openshift-kube-apiserver $i -c kube-apiserver -- \
        find /etc/kubernetes/static-pod-resources -type d -wholename
'*/secrets*' -exec stat -c %U:%G {} \;
        oc exec -n openshift-kube-apiserver $i -c kube-apiserver -- \
        find /etc/kubernetes/static-pod-resources -type f -wholename
```

```
'*/secrets*' -exec stat -c %U:%G {} \;
done
```

Verify that the ownership of all files and directories in this hierarchy is set to `root:root`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, the `static-pod-resources/secrets` and `static-pod-certs` directories and all of the files and directories contained within it, are set to be owned by the root user.

References:

1. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

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.

1.1.20 Ensure that the OpenShift PKI certificate file permissions are set to 644 or more restrictive (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that OpenShift PKI certificate files have permissions of `644` or more restrictive.

Rationale:

OpenShift makes use of a number of certificate files as part of the operation of its components. The permissions on these files should be set to 644 or more restrictive to protect their integrity.

Impact:

None

Audit:

Certificates for control plane components like `kube-apiserver`, `kube-controller-manager`, and `kube-scheduler` are stored in the directory `/etc/kubernetes/static-pod-certs/secrets`. Certificate files all have permissions `600`.
Run the following command.

```
# Should 644 or more restrictive

for i in $(oc -n openshift-kube-apiserver get pod -l app=openshift-kube-
apiserver -o jsonpath='{.items[*].metadata.name}')
do
    echo $i static-pod-certs
    oc exec -n openshift-kube-apiserver $i -c kube-apiserver -- \
    find /etc/kubernetes/static-pod-certs -type f -wholename
'*/secrets/*.crt' -exec stat -c %a {} \;
done
```

Verify that the permissions are `600`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, the certificates used by OpenShift are set to have permissions of `600`.

References:

1. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

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.

1.1.21 Ensure that the OpenShift PKI key file permissions are set to 600 (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that the OpenShift PKI key files have permissions of `600`.

Rationale:

OpenShift makes use of a number of key files as part of the operation of its components. The permissions on these files should be set to `600` to protect their integrity and confidentiality.

Impact:

None

Audit:

Keys for control plane components like `kube-apiserver`, `kube-controller-manager`, `ube-scheduler` and `etcd` are stored with their respective static pod configurations in the directory `/etc/kubernetes/static-pod-certs/secrets`. Key files all have permissions `600`.
Run the following command.

```
for i in $(oc -n openshift-kube-apiserver get pod -l app=openshift-kube-
apiserver -o jsonpath='{.items[*].metadata.name}')
do
    echo $i static-pod-certs
    oc exec -n openshift-kube-apiserver $i -c kube-apiserver -- \
    find /etc/kubernetes/static-pod-certs -type f -wholename
'*/secrets/*.key' -exec stat -c %a {} \;
done
```

Verify that the permissions are `600`.

Remediation:

No remediation required; file permissions are managed by the operator.

Default Value:

By default, the keys used by OpenShift are set to have permissions of `600`

References:

1. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

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.

1.2 API Server

This section contains recommendations relating to API server configuration flags.

OpenShift includes two API servers, the OpenShift API server and the Kubernetes API server. All API calls are directed to the Open Shift API server and then Kubernetes objects are delegated to `kube-apiserver`.

The cluster configuration resource (CR) APIServer holds configuration settings (like serving certificates, client CA and CORS domains) shared by all API servers in the system, among them especially `kube-apiserver` and `openshift-apiserver`. The canonical name of an instance is 'cluster'. Changes to the API server configurations should be done in the APIServer custom resource definition (CRD): apiservers.config.openshift.io

The OpenShift API Server is managed by the `openshift-apiserver-operator`

The Kubernetes API Server is managed by the `openshift-kube-apiserver-operator`

Both are managed by the Cluster Version Operator

1.2.1 Ensure that anonymous requests are authorized (Manual)

Profile Applicability:

- Level 1

Description:

When anonymous requests to the API server are allowed, they must be authorized.

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 API server. You should rely on authentication to authorize anonymous requests.

If you are using RBAC authorization, it is generally considered reasonable to allow anonymous access to the API Server for health checks and discovery purposes, and hence this recommendation is not scored. However, you should consider whether anonymous discovery is an acceptable risk for your purposes.

Impact:

Anonymous requests are assigned to the `system:unauthenticated` group which allows the system to determine which actions are allowed.

Audit:

OpenShift allows anonymous requests (then authorizes them). OpenShift allows anonymous requests to the API server to support information discovery and `webhook` integrations. OpenShift provides it's own fully integrated authentication and authorization mechanism. If no access token or certificate is presented, the authentication layer assigns the `system:anonymous` virtual user and the `system:unauthenticated` virtual group to the request. This allows the authorization layer to determine which requests, if any, an anonymous user is allowed to make.

```
# To verify that userGroups include system:unauthenticated
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq '.auditConfig.policyConfiguration.rules[]'

# To verify that userGroups include system:unauthenticated
oc get configmap config -n openshift-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq '.auditConfig.policyConfiguration.rules[]'

# To verify RBAC is enabled
oc get clusterrolebinding
oc get clusterrole
oc get rolebinding
oc get role
```

Verify that the userGroups include `system:unauthenticated`.
Verify that role bindings and roles are returned.

Remediation:

None required. The default configuration should not be modified.

Default Value:

By default, anonymous access is enabled and assigned to the `system:unauthenticated group`, which allows the system to determine which actions are allowed.

If the default behavior is changed, platform components will not work properly, in particular Elasticsearch and Prometheus. The `oauth-proxy` deployed as part of these components makes anonymous use of `/.well-known/oauth-authorization-server` endpoint, granted by `system:discovery` role.

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/understanding-authentication.html
2. https://docs.openshift.com/container-platform/4.5/authentication/using-rbac.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-authentication-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
5. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
7. https://kubernetes.io/docs/reference/access-authn-authz/authentication/#anonymous-requests

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

1.2.2 Ensure that the --basic-auth-file argument is not set (Automated)

Profile Applicability:

- Level 1

Description:

Do not use basic authentication.

Rationale:

Basic authentication uses plaintext credentials for authentication. Currently, the basic authentication credentials last indefinitely, and the password cannot be changed without restarting the API server. The basic authentication is currently supported for convenience. Hence, basic authentication should not be used.

Impact:

OpenShift uses tokens and certificates for authentication.

Audit:

OpenShift provides it's own fully integrated authentication and authorization mechanism. The `apiserver` is protected by either requiring an OAuth token issued by the platform's integrated OAuth server or signed certificates. The `basic-auth-file` method is not enabled in OpenShift.
Run the following command:

```
oc -n openshift-kube-apiserver get cm config -o yaml | grep --color "basic-
auth"
oc -n openshift-apiserver get cm config -o yaml | grep --color "basic-auth"
oc get clusteroperator authentication
```

Verify that the `--basic-auth-file` argument does not exist.
Verify that the `authentication-operator` is running: Available is True.

Remediation:

None required. `--basic-auth-file` cannot be configured on OpenShift.

Default Value:

By default, `--basic-auth-file` argument is not set and OAuth authentication is configured.

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/configuring-internal-oauth.html
2. https://docs.openshift.com/container-platform/4.5/authentication/understanding-authentication.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-authentication-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
5. https://kubernetes.io/docs/reference/access-authn-authz/authentication/#static-password-file

CIS Controls:

Version 6

16.14 Encrypt/Hash All Authentication Files And Monitor Their Access

Verify that all authentication files are encrypted or hashed and that these files cannot be accessed without root or administrator privileges. Audit all access to password files in the system.

Version 7

16.4 Encrypt or Hash all Authentication Credentials

Encrypt or hash with a salt all authentication credentials when stored.

1.2.3 Ensure that the --token-auth-file parameter is not set (Automated)

Profile Applicability:

- Level 1

Description:

Do not use token based authentication.

Rationale:

The token-based authentication utilizes static tokens to authenticate requests to the `apiserver`. The tokens are stored in clear-text in a file on the `apiserver`, and cannot be revoked or rotated without restarting the `apiserver`. Hence, do not use static token-based authentication.

Impact:

OpenShift does not use the `token-auth-file` flag. OpenShift includes a built-in OAuth server rather than relying on a static token file. The OAuth server is integrated with the API server.

Audit:

OpenShift does not use the token-auth-file flag. OpenShift includes a built-in OAuth server rather than relying on a static token file. Authentication is managed by the OpenShift `authentication-operator`. To verify that the `token-auth-file` flag is not present and that the `authentication-operator` is running, run the following commands:

```
# Verify that the token-auth-file flag is not present
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'
oc get configmap config -n openshift-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'
oc get kubeapiservers.operator.openshift.io cluster -o json | jq
'.spec.observedConfig.apiServerArguments'

#Verify that the authentication operator is running
oc get clusteroperator authentication
```

Verify that the `--token-auth-file` argument does not exist.
Verify that the `authentication-operator` is running: Available is True.

Remediation:

None is required.

Default Value:

By default, `--token-auth-file argument` is not set and OAuth authentication is configured.

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/configuring-internal-oauth.html
2. https://docs.openshift.com/container-platform/4.5/authentication/understanding-authentication.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-authentication-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
5. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
6. https://kubernetes.io/docs/reference/access-authn-authz/authentication/#static-token-file
7. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

CIS Controls:

Version 6

16.14 Encrypt/Hash All Authentication Files And Monitor Their Access

Verify that all authentication files are encrypted or hashed and that these files cannot be accessed without root or administrator privileges. Audit all access to password files in the system.

Version 7

16.4 Encrypt or Hash all Authentication Credentials

Encrypt or hash with a salt all authentication credentials when stored.

1.2.4 Use https for kubelet connections (Automated)

Profile Applicability:

- Level 1

Description:

Use https for kubelet connections.

Rationale:

Connections from `apiserver` to `kubelets` could potentially carry sensitive data such as secrets and keys. It is thus important to use in-transit encryption for any communication between the `apiserver` and `kubelets`.

Impact:

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

Audit:

OpenShift does not use the `--kubelet-https` argument. OpenShift utilizes X.509 certificates for authentication of the control-plane components. OpenShift configures the API server to use an internal certificate authority (CA) to validate the user certificate sent during TLS negotiation. If the validation of the certificate is successful, the request is authenticated and user information is derived from the certificate subject fields.
To verify the kubelet client certificates are present, run the following command:

```
#for 4.5
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.kubeletClientInfo'

#for 4.6
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'

#for both 4.5 and 4.6
oc -n openshift-apiserver describe secret serving-cert
```

Verify that the kubelet client-certificate and kubelet client-key files are present.
client-certificate:
`/etc/kubernetes/static-pod-resources/secrets/kubelet-client/tls.crt`
client-key:
`/etc/kubernetes/static-pod-resources/secrets/kubelet-client/tls.key`

Verify that the serving-cert for the `openshift-apiserver` is type kubernetes.io/tls and that returned Data includes `tls.crt` and `tls.key`.

Remediation:

No remediation is required. OpenShift platform components use X.509 certificates for authentication. OpenShift manages the CAs and certificates for platform components. This is not configurable.

Default Value:

By default, kubelet connections are encrypted.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
3. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L12-L13
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L94-L98
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/

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.

1.2.5 Ensure that the kubelet uses certificates to authenticate (Automated)

Profile Applicability:

- Level 1

Description:

Enable certificate based kubelet authentication.

Rationale:

The apiserver, by default, does not authenticate itself to the kubelet's HTTPS endpoints. The requests from the apiserver are treated anonymously. You should set up certificate-based kubelet authentication to ensure that the apiserver authenticates itself to kubelets when submitting requests.

Impact:

Require TLS to be configured on the apiserver as well as kubelets.

Audit:

OpenShift does not use the `--kubelet-client-certificate` or the `kubelet-client-key` arguments. OpenShift utilizes X.509 certificates for authentication of the control-plane components. OpenShift configures the API server to use an internal certificate authority (CA) to validate the user certificate sent during TLS negotiation. If the CA validation of the certificate is successful, the request is authenticated and user information is derived from the certificate subject fields.
To verify the certificates are present, run the following command:

```
#for 4.5
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.kubeletClientInfo'

#for 4.6
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'

#for both 4.5 and 4.6
oc -n openshift-apiserver describe secret serving-cert
```

Verify that the kubelet client-certificate and kubelet client-key files are present.
client-certificate:
`/etc/kubernetes/static-pod-resources/secrets/kubelet-client/tls.crt`

client-key:
`/etc/kubernetes/static-pod-resources/secrets/kubelet-client/tls.key`
Verify that the serving-cert for the `openshift-apiserver` is type `kubernetes.io/tls` and that returned Data includes `tls.crt` and `tls.key`.

Remediation:

No remediation is required. OpenShift platform components use X.509 certificates for authentication. OpenShift manages the CAs and certificates for platform components. This is not configurable.

Default Value:

By default, kubelet authentication is managed with X.509 certificates.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
3. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L12-L13
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L94-L98
5. https://kubernetes.io/docs/admin/kube-apiserver/
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/
7. https://kubernetes.io/docs/concepts/architecture/control-plane-node-communication/

CIS Controls:

Version 6

3.4 Use Only Secure Channels For Remote System Administration

Perform all remote administration of servers, workstation, network devices, and similar equipment over secure channels. Protocols such as telnet, VNC, RDP, or others that do not actively support strong encryption should only be used if they are performed over a secondary encryption channel, such as SSL, TLS or IPSEC.

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

Use multi-factor authentication and encrypted channels for all administrative account access.

1.2.6 Verify that the kubelet certificate authority is set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Verify kubelet's certificate before establishing connection.

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.

Impact:

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

Audit:

OpenShift does not use the `--kubelet-certificate-authority` flag. OpenShift utilizes X.509 certificates for authentication of the control-plane components. OpenShift configures the API server to use an internal certificate authority (CA) to validate the user certificate sent during TLS negotiation. If the CA validation of the certificate is successful, the request is authenticated and user information is derived from the certificate subject fields.
To verify, run the following command:

```
# for 4.5
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.kubeletClientInfo'

# for 4.6
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'
```

Verify that the value for ca is the following.
`"ca": "/etc/kubernetes/static-pod-resources/configmaps/kubelet-serving-ca/ca-bundle.crt"`

Remediation:

No remediation is required. OpenShift platform components use X.509 certificates for authentication. OpenShift manages the CAs and certificates for platform components. This is not configurable.

Default Value:

By default, kubelet authentication is managed with X.509 certificates.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/
5. https://kubernetes.io/docs/concepts/architecture/control-plane-node-communication/

CIS Controls:

Version 6

3.4 Use Only Secure Channels For Remote System Administration

Perform all remote administration of servers, workstation, network devices, and similar equipment over secure channels. Protocols such as telnet, VNC, RDP, or others that do not actively support strong encryption should only be used if they are performed over a secondary encryption channel, such as SSL, TLS or IPSEC.

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

Use multi-factor authentication and encrypted channels for all administrative account access.

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

Profile Applicability:

- Level 1

Description:

Do not always authorize all requests.

Rationale:

The API Server, can be configured to allow all requests. This mode should not be used on any production cluster.

Impact:

Only authorized requests will be served.

Audit:

It is not possible to configure an OpenShift cluster to allow all requests. OpenShift is configured at bootstrap time to use RBAC to authorize requests. Role-based access control (RBAC) objects determine what actions a user is allowed to perform on what objects in an OpenShift cluster. Cluster administrators manage RBAC for the cluster. Project owners can manage RBAC for their individual OpenShift projects. The OpenShift API server configmap does not use the `authorization-mode` flag.
To verify, run the following commands:

```
# To verify that the authorization-mode argument is not used
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'
oc get configmap config -n openshift-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'

# To verify RBAC is configured:
oc get clusterrolebinding
oc get clusterrole
oc get rolebinding
oc get role
```

For OPC 4.5 and earlier, verify that the `authorization-mode` argument does not appear in the output. Verify the expected roles and role bindings are returned.
For OCP 4.6 and above, verify that the `authorization-mode` argument includes RBAC.

Remediation:

None. RBAC is always on and the OpenShift API server does not use the values assigned to the flag authorization-mode.

Default Value:

OpenShift uses RBAC by default.

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/using-rbac.html
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://kubernetes.io/docs/admin/kube-apiserver/
5. https://kubernetes.io/docs/reference/access-authn-authz/authorization/

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.

1.2.8 Verify that the Node authorizer is enabled (Automated)

Profile Applicability:

- Level 1

Description:

Restrict kubelet nodes to reading only objects associated with them.

Rationale:

The Node authorization mode only allows kubelets to read `Secret`, `ConfigMap`, `PersistentVolume`, and `PersistentVolumeClaim` objects associated with their nodes.

Impact:

None

Audit:

In OpenShift, the Node authorizer is enabled by default and is not configurable. In OpenShift 4.5 and earlier the OpenShift API server `configmap` does not use the `authorization-mode` flag.

```
# For OCP 4.5 and earlier verify that authorization-mode is not used
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'
oc get kubeapiservers.operator.openshift.io cluster -o json | jq
'.spec.observedConfig.apiServerArguments'

# For OCP 4.5 and earlier verify that authorization-mode is not used
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host cat /etc/kubernetes/kubelet.conf
| grep authorization-mode
        oc debug node/${node} -- chroot /host ps -aux | grep kubelet | grep
authorization-mode
done

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'
```

For OCP 4.5, verify that the `authorization-mode` argument is not present.
For OCP 4.6 and above, verify that the `authorization-mode` argument includes `Node`.
Verify the no overrides are configured.

Remediation:

No remediation is required.

Default Value:

By default, in OpenShift 4.5 and earlier, the Node authorizer is compiled into the API server and is not visible. In OpenShift 4.6, `authorization-mode` includes Node by default.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
4. https://kubernetes.io/docs/reference/access-authn-authz/node/
5. https://github.com/kubernetes/kubernetes/pull/46076
6. https://acotten.com/post/kube17-security

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.

1.2.9 Verify that RBAC is enabled (Automated)

Profile Applicability:

- Level 1

Description:

Turn on Role Based Access Control.

Rationale:

Role Based Access Control (RBAC) allows fine-grained control over the operations that different entities can perform on different objects in the cluster. It is recommended to use the RBAC authorization mode.

Impact:

When RBAC is enabled you will need to ensure that appropriate RBAC settings (including Roles, RoleBindings and ClusterRoleBindings) are configured to allow appropriate access.

Audit:

OpenShift is configured at bootstrap time to use RBAC to authorize requests. Role-based access control (RBAC) objects determine what actions a user is allowed to perform on what objects in an OpenShift cluster. Cluster administrators manage RBAC for the cluster. Project owners can manage RBAC for their individual OpenShift projects. The OpenShift API server `configmap` does not use the `authorization-mode` flag.
To verify, run the following commands:

```
# For 4.5 To verify that the authorization-mode argument is not used
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'
oc get configmap config -n openshift-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'

# To verify RBAC is used
oc get clusterrolebinding
oc get clusterrole
oc get rolebinding
oc get role

# For 4.6, verify that the authorization-mode argument includes RBAC
```

```
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments["authorization-mode"]'
```

For OCP 4.5, verify that the `authorization-mode` argument is not present. Verify the expected roles and role bindings are returned.
For OCP 4.6 and above, verify that the `authorization-mode` argument includes RBAC.
Verify the no overrides are configured.

Remediation:

None. It is not possible to disable RBAC.

Default Value:

OpenShift uses RBAC by default. OpenShift includes default roles and role bindings. Custom roles and role bindings can be added.

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/using-rbac.html
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-authentication-operator_red-hat-operators
5. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/bootkube/manifests/cluster-role-binding-kube-apiserver.yaml
6. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L17-L21
7. https://kubernetes.io/docs/reference/access-authn-authz/rbac/

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.

1.2.10 Ensure that the APIPriorityAndFairness feature gate is enabled (Manual)

Profile Applicability:

- Level 1

Description:

Limit the rate at which the API server accepts requests.

Rationale:

Using `EventRateLimit` admission control enforces a limit on the number of events that the API Server will accept in a given time slice. A misbehaving workload could overwhelm and DoS the API Server, making it unavailable. This particularly applies to a multi-tenant cluster, where there might be a small percentage of misbehaving tenants which could have a significant impact on the performance of the cluster overall. Hence, it is recommended to limit the rate of events that the API server will accept.

Note: This is an Alpha feature in the Kubernetes 1.15 release.

Impact:

None, as the OpenShift kubelet has been fixed to send fewer requests.

Audit:

EventRateLimit admission plugin cannot be enabled in Openshift. It was developed to alleviate the potential issue of flooding the API server with requests. However, the kubelet has since been fixed to send fewer events. OpenShift 4.5 and forward uses the api priority and fairness feature to limit the rate at which the API server accepts requests.
Run the following command:

```
#Verify the APIPriorityAndFairness feature-gate
oc get kubeapiservers.operator.openshift.io cluster -o json | jq
'.spec.observedConfig.apiServerArguments'

#Verify the set of admission-plugins for OCP 4.6 and higher
oc -n openshift-kube-apiserver get configmap config -o json | jq -r
'.data."config.yaml"' | jq '.apiServerArguments."enable-admission-plugins"'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'
```

For 4.5, verify that the feature-gate is turned on for the APIServer priority and fairness: `APIPriorityAndFairness=true`. In OCP 4.5 and earlier, the default set of admission plugins are compiled into the `apiserver` and are not visible in the configuration yaml.
For 4.6 and above, verify that the set of admission plugins does not include `EventRateLimit`.
Verify that no unsupported Overrides are configured.

Remediation:

No remediation is required.

Default Value:

By default, the OpenShift kubelet has been fixed to send fewer requests.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/admission-plug-ins.html
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L34-L78
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
6. https://kubernetes.io/docs/concepts/cluster-administration/flow-control/
7. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#eventratelimit
8. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#eventratelimit
9. https://github.com/staebler/community/blob/9873b632f4d99b5d99c38c9b15fe2f8b93d0a746/contributors/design-proposals/admission_control_event_rate_limit.md

CIS Controls:

Version 6

8.4 Enable Anti-exploitation Features (i.e. DEP, ASLR, EMET)
Enable anti-exploitation features such as Data Execution Prevention (DEP), Address Space Layout Randomization (ASLR), virtualization/containerization, etc. For increased protection, deploy capabilities such as Enhanced Mitigation Experience Toolkit (EMET)

that can be configured to apply these protections to a broader set of applications and executables.

Version 7

8.3 Enable Operating System Anti-Exploitation Features/ Deploy Anti-Exploit Technologies

Enable anti-exploitation features such as Data Execution Prevention (DEP) or Address Space Layout Randomization (ASLR) that are available in an operating system or deploy appropriate toolkits that can be configured to apply protection to a broader set of applications and executables.

1.2.11 Ensure that the admission control plugin AlwaysAdmit is not set (Automated)

Profile Applicability:

- Level 1

Description:

Do not allow all requests.

Rationale:

Setting admission control plugin `AlwaysAdmit` allows all requests and does not filter any requests.

The `AlwaysAdmit` admission controller was deprecated in Kubernetes v1.13. Its behavior was equivalent to turning off all admission controllers.

Impact:

Only requests explicitly allowed by the admissions control plugins would be served.

Audit:

This controller is disabled by default in OpenShift and cannot be enabled. It has also been deprecated by the Kubernetes community as it behaves as if there were no controller. Run the following command:

```
#Verify the set of admission-plugins for OCP 4.6 and higher
oc -n openshift-kube-apiserver get configmap config -o json | jq -r
'.data."config.yaml"' | jq '.apiServerArguments."enable-admission-plugins"'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'
```

In OCP 4.5 and earlier, the default set of admission plugins are compiled into the `apiserver` and are not visible in the configuration yaml.
For 4.6 and above, verify that the set of admission plugins does not include `AlwaysAdmit`.
Verify that no unsupported Overrides are configured.

Remediation:

No remediation is required. The `AlwaysAdmit` admission controller cannot be enabled in OpenShift.

Default Value:

This `AlwaysAdmit` controller is disabled by default in OpenShift and cannot be enabled.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/admission-plug-ins.html
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L34-L78
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
6. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#alwaysadmit

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

1.2.12 Ensure that the admission control plugin AlwaysPullImages is not set (Manual)

Profile Applicability:

- Level 1

Description:

Always pull images.

Rationale:

Setting admission control policy to `AlwaysPullImages` forces every new pod to pull the required images every time. In a multi-tenant cluster users can be assured that their private images can only be used by those who have the credentials to pull them. Without this admission control policy, once an image has been pulled to a node, any pod from any user can use it simply by knowing the image's name, without any authorization check against the image ownership. When this plug-in is enabled, images are always pulled prior to starting containers, which means valid credentials are required.

However, turning on this admission plugin can introduce new kinds of cluster failure modes. OpenShift 4 master and infrastructure components are deployed as pods. Enabling this feature can result in cases where loss of contact to an image registry can cause a redeployed infrastructure pod (`oauth-server` for example) to fail on an image pull for an image that is currently present on the node. We use `PullIfNotPresent` so that a loss of image registry access does not prevent the pod from starting. If it becomes `PullAlways`, then an image registry access outage can cause key infrastructure components to fail.

This can be managed per container. When OpenShift Container Platform creates containers, it uses the container's `imagePullPolicy` to determine if the image should be pulled prior to starting the container. There are three possible values for `imagePullPolicy`: `Always`, `IfNotPresent`, `Never`. If a container's `imagePullPolicy` parameter is not specified, OpenShift Container Platform sets it based on the image's tag. If the tag is latest, OpenShift Container Platform defaults `imagePullPolicy` to `Always`. Otherwise, OpenShift Container Platform defaults `imagePullPolicy` to `IfNotPresent`.

Impact:

Credentials would be required to pull the private images every time. Also, in trusted environments, this might increases load on network, registry, and decreases speed.

This setting could impact offline or isolated clusters, which have images pre-loaded and do not have access to a registry to pull in-use images. This setting is not appropriate for clusters which use this configuration.

Audit:

Run the following command

```
#Verify the set of admission-plugins for OCP 4.6 and higher
oc -n openshift-kube-apiserver get configmap config -o json | jq -r
'.data."config.yaml"' | jq
'.apiServerArguments."enable-admission-plugins"'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'
```

In OCP 4.5 and earlier, the default set of admission plugins are compiled into the `apiserver` and are not visible in the configuration yaml.
For 4.6 and above, verify that the set of admission plugins does not include `AlwaysPullImages`.
Verify that no unsupported Overrides are configured

Remediation:

None required.

Default Value:

When OpenShift Container Platform creates containers, it uses the container's `imagePullPolicy` to determine if the image should be pulled prior to starting the container.

References:

1. https://docs.openshift.com/container-platform/4.5/openshift_images/managing_images/image-pull-policy.html
2. https://docs.openshift.com/container-platform/4.5/architecture/admission-plug-ins.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L34-L78
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

7. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#alwayspullimages

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected 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.

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.

1.2.13 Ensure that the admission control plugin SecurityContextDeny is not set (Manual)

Profile Applicability:

- Level 1

Description:

The `SecurityContextDeny` admission controller can be used to deny pods which make use of some `SecurityContext` fields which could allow for privilege escalation in the cluster. This should be used where `PodSecurityPolicy` is not in place within the cluster.

Rationale:

`SecurityContextDeny` can be used to provide a layer of security for clusters which do not have `PodSecurityPolicies` enabled.

Impact:

The `SecurityContextDeny` admission controller cannot be enabled as it conflicts with the `SecurityContextConstraint` admission controller.

Audit:

In OpenShift, RBAC roles can restrict access to pod subresources 'exec' and 'attach', while Security Context Constraints (SCCs) restrict access to run privileged containers. By default, OpenShift runs pods on worker nodes as unprivileged (with the restricted SCC). Unlike upstream Kubernetes, OpenShift does not enable the `SecurityContextDeny` admission control plugin. OpenShift's Security Context Constraint (SCC) admission controller provides the same level of restriction as Pod Security Policy (PSP). PSPs are still beta in Kubernetes. Run the following commands:

```
#Verify the set of admission-plugins for OCP 4.6 and higher
oc -n openshift-kube-apiserver get configmap config -o json | jq -r
'.data."config.yaml"' | jq '.apiServerArguments."enable-admission-plugins"'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'

#Verify that SecurityContextConstraints are deployed
oc get scc
oc describe scc restricted
```

Verify that the list of admission controllers does not include `SecurityContextDeny`.
For 4.6 and above, verify that the set of admission plugins includes `SecurityContextConstraint` and that the `--disable-admission-plugins` argument is set to a value that does not include `SecurityContextConstraint`. In OCP 4.5 and earlier, the default set of admission plugins are compiled into the `apiserver` and are not visible in the configuration yaml. and does include.
Verify that no unsupported Overrides are configured.
Verify that the list of SCCs includes `anyuid`, `hostaccess`, `hostmount-anyuid`, `hostnetwork`, `node-exporter`, `nonroot`, `privileged`, `restricted`
Verify that the restricted SCC sets `Allowed Privileged` to `false`.

Remediation:

None required. The Security Context Constraint admission controller cannot be disabled in OpenShift 4.

Default Value:

By default, OpenShift uses Security Context Constraints (SCCs) to restrict access to run privileged containers and runs pods on worker nodes as unprivileged (with the restricted SCC).

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://docs.openshift.com/container-platform/4.5/architecture/admission-plug-ins.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L34-L78
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
7. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#securitycontextdeny
8. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#working-with-rbac

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.

1.2.14 Ensure that the admission control plugin ServiceAccount is set (Automated)

Profile Applicability:

- Level 1

Description:

Automate service accounts management.

Rationale:

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. You should create your own service account and let the API server manage its security tokens.

Impact:

None.

Audit:

The ServiceAccount admission control plugin is enabled by default. Every service account has an associated user name that can be granted roles, just like a regular user. The user name for each service account is derived from its project and the name of the service account. Service accounts are required in each project to run builds, deployments, and other pods. The default service accounts that are automatically created for each project are isolated by the project namespace.
Run the following commands:

```
#Verify the list of admission controllers for 4.6 and higher
oc -n openshift-kube-apiserver get configmap config -o json | jq -r
'.data."config.yaml"' | jq '.apiServerArguments."enable-admission-plugins"'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'

#Verify that Service Accounts are present
oc get sa -A
```

For 4.6 and above, verify that the set of admission plugins does include `ServiceAccount` and that the `--disable-admission-plugins` argument is set to a value that does not include `ServiceAccount`.

In OCP 4.5 and earlier, the default set of admission plugins are compiled into the `apiserver` and are not visible in the configuration yaml. and does include.
Verify that no unsupported Overrides are configured.
Verify that Service Accounts are present for every namespace.

Remediation:

None required. OpenShift is configured to use service accounts by default.

Default Value:

By default, `ServiceAccount` is set.

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/understanding-and-creating-service-accounts.html
2. https://docs.openshift.com/container-platform/4.5/architecture/admission-plug-ins.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L34-L78
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
7. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#serviceaccount
8. https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

CIS Controls:

Version 6

16 Account Monitoring and Control
Account Monitoring and Control

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

1.2.15 Ensure that the admission control plugin NamespaceLifecycle is set (Automated)

Profile Applicability:

- Level 1

Description:

Reject creating objects in a namespace that is undergoing termination.

Rationale:

Setting admission control policy to `NamespaceLifecycle` ensures that objects cannot be created in non-existent namespaces, and that namespaces undergoing termination are not used for creating the new objects. This is recommended to enforce the integrity of the namespace termination process and also for the availability of the newer objects.

Impact:

None

Audit:

OpenShift enables the `NamespaceLifecycle` plugin by default.
Run the following command:

```
#Verify the list of admission controllers for 4.6 and higher
oc -n openshift-kube-apiserver get configmap config -o json | jq -r
'.data."config.yaml"' | jq '.apiServerArguments."enable-admission-plugins"'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'
```

In OCP 4.5 and earlier, the default set of admission plugins are compiled into the `apiserver` and are not visible in the configuration yaml.
For 4.6 and above, verify that the set of admission plugins includes `NamespaceLifecycle` and that the `--disable-admission-plugins` argument is set to a value that does not include `NamespaceLifecycle`.
Verify that no unsupported Overrides are configured.

Remediation:

Ensure that the `--disable-admission-plugins` parameter does not include `NamespaceLifecycle`.

Default Value:

By default, `NamespaceLifecycle` is set.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/admission-plug-ins.html
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L34-L78
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
6. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#namespacelifecycle

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.2.16 Ensure that the admission control plugin SecurityContextConstraint is set (Automated)

Profile Applicability:

- Level 1

Description:

Reject creating pods that do not match Pod Security Policies.

Rationale:

A Pod Security Policy is a cluster-level resource that controls the actions that a pod can perform and what it has the ability to access. The `PodSecurityPolicy` objects define a set of conditions that a pod must run with in order to be accepted into the system. Pod Security Policies are composed of settings and strategies that control the security features a pod has access to and hence this must be used to control pod access permissions.

Note: When the `PodSecurityPolicy` admission plugin is in use, there needs to be at least one `PodSecurityPolicy` in place for ANY pods to be admitted. See section 5.2 for recommendations on `PodSecurityPolicy` settings.

Impact:

Default Security Context Constraint objects are present on the cluster and granted by default based on roles. Custom SCCs can be created and granted as needed.

Audit:

OpenShift provides the same protection via the SecurityContextConstraints admission plugin, which is enabled by default. The `PodSecurityPolicy` admission control plugin is disabled by default as it is still beta and not yet supported with OpenShift. Security Context Constraints (SCCs) and Pod Security Policy cannot be used on the same cluster.
Run the following command:

```
#Verify the set of admission-plugins for OCP 4.6 and higher
oc -n openshift-kube-apiserver get configmap config -o json | jq -r
'.data."config.yaml"' | jq '.apiServerArguments."enable-admission-plugins"'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'

#Verify that SecurityContextConstraints are deployed
```

```
oc get scc
oc describe scc restricted
```

In OCP 4.5 and earlier, the default set of admission plugins are compiled into the `apiserver` and are not visible in the configuration yaml. and does include.
For 4.6 and above, verify that the set of admission plugins includes `SecurityContextConstraint` and that the `--disable-admission-plugins` argument is set to a value that does not include `SecurityContextConstraint`.
Verify that no unsupported Overrides are configured.
Verify that the list of SCCs includes `anyuid`, `hostaccess`, `hostmount-anyuid`, `hostnetwork`, `node-exporter`, `nonroot`, `privileged`, `restricted`
Verify that the restricted SCC sets `Allowed Privileged` to `false`.

Remediation:

None required. Security Context Constraints are enabled by default in OpenShift and cannot be disabled.

Default Value:

By default, `SecurityContextConstraints` admission controller is configured and cannot be disabled.

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://docs.openshift.com/container-platform/4.5/architecture/admission-plug-ins.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L34-L78
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
7. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#podsecuritypolicy
8. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.2.17 Ensure that the admission control plugin NodeRestriction is set (Automated)

Profile Applicability:

- Level 1

Description:

Limit the `Node` and `Pod` objects that a kubelet could modify.

Rationale:

Using the `NodeRestriction` plug-in ensures that the kubelet is restricted to the `Node` and `Pod` objects that it could modify as defined. Such kubelets will only be allowed to modify their own `Node` API object, and only modify `Pod` API objects that are bound to their node.

Impact:

None

Audit:

In OpenShift, the `NodeRestriction` admission plugin is enabled by default.
Run the following command:

```
# For 4.5, review the control plane manifest
https://github.com/openshift/origin/blob/release-
4.5/vendor/k8s.io/kubernetes/cmd/kubeadm/app/phases/controlplane/manifests.go
#L132

#Verify the set of admission-plugins for OCP 4.6 and higher
oc -n openshift-kube-apiserver get configmap config -o json | jq -r
'.data."config.yaml"' | jq '.apiServerArguments."enable-admission-plugins"'

#Check that no overrides are configured
oc get kubeapiservers.operator.openshift.io cluster -o json | jq -r
'.spec.unsupportedConfigOverrides'
```

In OCP 4.5 and earlier, the default set of admission plugins are compiled into the `apiserver` and are not visible in the configuration yaml.
For 4.6 and above, verify that the set of admission plugins includes `NodeRestriction` and that the `--disable-admission-plugins` argument is set to a value that does not include `NodeRestriction`.
Verify that no unsupported Overrides are configured.

Remediation:

The `NodeRestriction` plugin cannot be disabled.

Default Value:

In OpenShift, the `NodeRestriction` admission plugin is enabled by default.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/admission-plug-ins.html
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://github.com/openshift/origin/blob/release-4.5/vendor/k8s.io/kubernetes/cmd/kubeadm/app/phases/controlplane/manifests.go#L132

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.2.18 Ensure that the --insecure-bind-address argument is not set (Automated)

Profile Applicability:

- Level 1

Description:

Do not bind the insecure API service.

Rationale:

If you bind the `apiserver` to an insecure address, basically anyone who could connect to it over the insecure port, would have unauthenticated and unencrypted access to your master node. The `apiserver` doesn't do any authentication checking for insecure binds and traffic to the Insecure API port is not encrypted, allowing attackers to potentially read sensitive data in transit.

Impact:

Connections to the API server will require valid authentication credentials.

Audit:

The `openshift-kube-apiserver` is served over HTTPS with authentication and authorization; the secure API endpoint for the `penshift-kube-apiserver` is bound to `0.0.0.0:6443` by `default`. Note that the `openshift-apiserver` is not running in the host network namespace. The port is not exposed on the node, but only through the pod network.
Run the following command:

```
# InsecureBindAddress=true should not be in the results
oc get kubeapiservers.operator.openshift.io cluster -o jsonpath='{range
.spec.observedConfig.apiServerArguments.feature-gates[*]}{@}{"\n"}{end}'

# Result should be only 6443
oc -n openshift-kube-apiserver get endpoints -o
jsonpath='{.items[*].subsets[*].ports[*].port}'

# Result should be only 8443
oc -n openshift-apiserver get endpoints -o
jsonpath='{.items[*].subsets[*].ports[*].port}'
```

Verify that the `--insecure-bind-address` argument does not exist.
Verify that the `bindAddress` is set to `6443` for the `openshift-kube-apiserver` and `8443` for

the `openshift-apiserver`.
Note that the `openshift-apiserver` is not running in the host network namespace. The port is not exposed on the node, but only through the pod network.

Remediation:

None required.

Default Value:

By default, the `openshift-kube-apiserver` is served over HTTPS with authentication and authorization; the secure API endpoint is `bound` to `0.0.0.0:6443`. Note that the `openshift-apiserver` is not running in the host network namespace. The port is not exposed on the node, but only through the pod network.

References:

1. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L104-L105
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

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.

1.2.19 Ensure that the --insecure-port argument is set to 0 (Automated)

Profile Applicability:

- Level 1

Description:

Do not bind to insecure port.

Rationale:

Setting up the `apiserver` to serve on an insecure port would allow unauthenticated and unencrypted access to your master node. This would allow attackers who could access this port, to easily take control of the cluster.

Impact:

All components that use the API must connect via the secured port, authenticate themselves, and be authorized to use the API.

This includes:

- kube-controller-manager
- kube-proxy
- kube-scheduler
- kubelets

Audit:

The `openshift-kube-apiserver` is served over HTTPS with authentication and authorization; the secure API endpoint is bound to `0.0.0.0:6443` by default. By default the `insecure-port` argument is set to `0`. Note that the `openshift-apiserver` is not running in the host network namespace. The port is not exposed on the node, but only through the pod network.
Run the following command:

```
# Should return 6443
oc -n openshift-kube-apiserver get endpoints -o
jsonpath='{.items[*].subsets[*].ports[*].port}'

# For OCP 4.6 and above
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments["insecure-port"]'
```

Verify that the `openshift-kube-apiserver` container-port is set to `6443`.
For OCP 4.6 and above, verify that `--insecure-port` argument in the API server operator configs is set to `0`.
Note that the parameter `kube-apiserver-insecure-readyz` has the argument `insecure-port` set to `6080`. This value should not be changed.

Remediation:

None required. The configuration is managed by the API server operator.

Default Value:

By default, the `openshift-kube-server` is served over HTTPS with authentication and authorization; the secure API endpoint is bound to `0.0.0.0:6443` and the `insecure-port` is set to `0`. Note that the `openshift-apiserver` is not running in the host network namespace. The port is not exposed on the node, but only through the pod network.

References:

1. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L102-L103
2. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L103-L105
3. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/kube-apiserver/pod.yaml#L155-L157
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
5. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

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.

1.2.20 Ensure that the --secure-port argument is not set to 0 (Automated)

Profile Applicability:

- Level 1

Description:

Do not disable the secure port.

Rationale:

The secure port is used to serve https with authentication and authorization. If you disable it, no https traffic is served and all traffic is served unencrypted.

Impact:

You need to set the API Server up with the right TLS certificates.

Audit:

The `openshift-kube-apiserver` is served over HTTPS with authentication and authorization; the secure API endpoint is bound to `0.0.0.0:6443` by `default`. In OpenShift, the only supported way to access the API server pod is through the load balancer and then through the internal service. The value is set by the `bindAddress` argument under the `servingInfo` parameter.
Run the following command:

```
oc get kubeapiservers.operator.openshift.io cluster -o json | jq
'.spec.observedConfig'

# Should return only 6443
oc get pods -n openshift-kube-apiserver -l app=openshift-kube-apiserver -o
jsonpath='{.items[*].spec.containers[?(@.name=="kube-
apiserver")].ports[*].containerPort}'
```

Verify that the `--bindAddress` argument is set `0.0.0.0:6443`.
Verify that the only port returned is `6443`.

Remediation:

None required.

Default Value:

By default, the `openshift-kube-apiserver` is served over HTTPS with authentication and authorization; the secure API `endpoint` is bound to `0.0.0.0:6443`. Note that the `openshift-apiserver` is not running in the host network namespace. The port is not exposed on the node, but only through the pod network.

The OpenShift platform manages the TLS certificates for the API servers. External access is only available through the load balancer and then through the internal service.

References:

1. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L102-L103
2. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L103-L105
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/

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.

1.2.21 Ensure that the healthz endpoint is protected by RBAC (Automated)

Profile Applicability:

- Level 1

Description:

Disable profiling, if not needed.

Rationale:

Profiling allows for the identification of specific performance bottlenecks. It generates a significant amount of program data that could potentially be exploited to uncover system and program details. If you are not experiencing any bottlenecks and do not need the profiler for troubleshooting purposes, it is recommended to turn it off to reduce the potential attack surface.

Impact:

Profiling information would not be available.

Audit:

Profiling is enabled by default in OpenShift. The API server operators expose Prometheus metrics via the metrics service. Profiling data is sent to `healthzPort`, the port of the localhost `healthz` endpoint. Changing this value may disrupt components that monitor the kubelet health. The default port value is 10248, and the `healthz BindAddress` is `127.0.0.1`
To ensure the collected data is not exploited, profiling endpoints are exposed at each master port and secured via RBAC (see cluster-debugger role). By default, the profiling endpoints are accessible only by users bound to `cluster-admin` or `cluster-debugger` role. Profiling cannot be disabled.
To verify the configuration, run the following command:

```
# Verify endpoints
oc -n openshift-kube-apiserver describe endpoints

# Check config for ports, livenessProbe, readinessProbe, healthz
oc -n openshift-kube-apiserver get cm kube-apiserver-pod -o json | jq -r
'.data."pod.yaml"' | jq '.spec.containers'

# Test to validate RBAC enabled on the apiserver endpoint; check with non-
admin role
```

```
oc project openshift-kube-apiserver

POD=$(oc get pods -n openshift-kube-apiserver -l app=openshift-kube-apiserver
-o jsonpath='{.items[0].metadata.name}')

PORT=$(oc get pods -n openshift-kube-apiserver -l app=openshift-kube-
apiserver -o jsonpath='{.items[0].spec.containers[0].ports[0].hostPort}')

# Following should return 403 Forbidden
oc rsh -n openshift-kube-apiserver ${POD} curl
https://localhost:${PORT}/metrics -k

# Create a service account to test RBAC
oc create -n openshift-kube-apiserver sa permission-test-sa

# Should return 403 Forbidden
SA_TOKEN=$(oc sa -n openshift-kube-apiserver get-token permission-test-sa)
oc rsh -n openshift-kube-apiserver ${POD} curl
https://localhost:${PORT}/metrics -H "Authorization: Bearer $SA_TOKEN" -k

# Cleanup
oc delete -n openshift-kube-apiserver sa permission-test-sa

# As cluster admin, should succeed
CLUSTER_ADMIN_TOKEN=$(oc whoami -t)
oc rsh -n openshift-kube-apiserver ${POD} curl
https://localhost:${PORT}/metrics -H "Authorization: Bearer
$CLUSTER_ADMIN_TOKEN" -k
```

Verify that the `livenessProbe` and `readinessProbe` are set to path: `healthz`.
Verify that only users with the `cluster_admin` role can retrieve metrics from the endpoint.

Remediation:

None required as profiling data is protected by RBAC.

Default Value:

By default, profiling is enabled.

References:

1. https://github.com/openshift/kubernetes-kubelet/blob/master/config/v1beta1/types.go#L259-L277
2. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/kube-apiserver/pod.yaml#L71-L84
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators

5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
6. https://github.com/kubernetes/community/blob/master/contributors/devel/profiling.md

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

1.2.22 Ensure that the --audit-log-path argument is set (Automated)

Profile Applicability:

- Level 1

Description:

Enable auditing on the Kubernetes API Server and set the desired audit log path.

Rationale:

Auditing the Kubernetes API Server provides a security-relevant chronological set of records documenting the sequence of activities that have affected the system by individual users, administrators or other components of the system. Even though currently, Kubernetes provides only basic audit capabilities, it should be enabled. You can enable it by setting an appropriate audit log path.

Impact:

None

Audit:

OpenShift audit works at the API server level, logging all requests coming to the server. API server audit is on by default.
Run the following command:

```
# Should return "/var/log/kube-apiserver/audit.log"
oc get configmap config -n openshift-kube-apiserver -o
jsonpath="{['.data.config\.yaml']}" | jq '.auditConfig.auditFilePath'

POD=$(oc get pods -n openshift-kube-apiserver -l app=openshift-kube-apiserver
-o jsonpath='{.items[0].metadata.name}')
oc rsh -n openshift-kube-apiserver -c kube-apiserver $POD ls /var/log/kube-
apiserver/audit.log

# Should return 0
echo $?

# Should return "/var/log/openshift-apiserver/audit.log"
oc get configmap config -n openshift-apiserver -o
jsonpath="{['.data.config\.yaml']}" | jq '.auditConfig.auditFilePath'

POD=$(oc get pods -n openshift-apiserver -l apiserver=true -o
jsonpath='{.items[0].metadata.name}')
oc rsh -n openshift-apiserver $POD ls /var/log/openshift-apiserver/audit.log
```

```
# Should return 0
echo $?
```

Verify that the `auditFilePath` is set to `/var/log/kube-apiserver/audit.log` for `openshift-kube-apiserver` and to `/var/log/openshift-apiserver/audit.log` for `openshift-apiserver`.

Remediation:

None required. This is managed by the cluster `apiserver` operator.

Default Value:

By default, auditing is enabled.

References:

1. https://docs.openshift.com/container-platform/4.5/nodes/nodes/nodes-nodes-audit-log.html
2. https://github.com/openshift/cluster-kube-apiserver-operator/blob/master/bindata/v4.1.0/config/defaultconfig.yaml#L22-L31
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
6. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
7. https://github.com/kubernetes/features/issues/22

CIS Controls:

Version 6

6.2 Ensure Audit Log Settings Support Appropriate Log Entry Formatting

Validate audit log settings for each hardware device and the software installed on it, ensuring that logs include a date, timestamp, source addresses, destination addresses, and various other useful elements of each packet and/or transaction. Systems should record logs in a standardized format such as syslog entries or those outlined by the Common Event Expression initiative. If systems cannot generate logs in a standardized format, log normalization tools can be deployed to convert logs into such a format.

Version 7

6.2 Activate audit logging

Ensure that local logging has been enabled on all systems and networking devices.

6.3 Enable Detailed Logging

Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.

1.2.23 Ensure that the audit logs are forwarded off the cluster for retention (Automated)

Profile Applicability:

- Level 1

Description:

Retain the logs for at least 30 days or as appropriate.

Rationale:

Retaining logs for at least 30 days ensures that you can go back in time and investigate or correlate any events. Set your audit log retention period to 30 days or as per your business requirements.

Impact:

None

Audit:

OpenShift audit works at the API server level, logging all requests coming to the server. Audit is on by default. Best practice is to ship audit logs off the cluster for retention. OpenShift includes the optional Cluster Logging operator and the `ElasticSearch` operator. OpenShift cluster logging can be configured to send logs to destinations outside of your OpenShift Container Platform cluster instead of the default `Elasticsearc`h log store using the following methods:

- Sending logs using the `Fluentd` forward protocol. You can create a `Configmap` to use the `Fluentd` forward protocol to securely send logs to an external logging aggregator that accepts the Fluent forward protocol.
- Sending logs using syslog. You can create a `Configmap` to use the syslog protocol to send logs to an external syslog (RFC 3164) server.

Alternatively, you can use the Log Forwarding API, currently in Technology Preview. The Log Forwarding API, which is easier to configure than the `Fluentd` protocol and syslog, exposes configuration for sending logs to the internal `lasticsearch` log store and to external `Fluentd` log aggregation solutions.
You cannot use the ConfigMap methods and the Log Forwarding API in the same cluster.
Verify that audit log forwarding is configured as appropriate.

Remediation:

Follow the documentation for log forwarding. Forwarding logs to third party systems

Default Value:

By default, auditing is enabled.

References:

1. https://access.redhat.com/solutions/4262201
2. https://docs.openshift.com/container-platform/4.5/logging/cluster-logging-external.html
3. https://docs.openshift.com/container-platform/4.5/nodes/nodes/nodes-nodes-audit-log.html
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L41-L77
5. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
6. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
7. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
8. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
9. https://github.com/kubernetes/features/issues/22

CIS Controls:

Version 6

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)

Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

Version 7

6.4 Ensure adequate storage for logs

Ensure that all systems that store logs have adequate storage space for the logs generated.

1.2.24 Ensure that the maximumRetainedFiles argument is set to 10 or as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Retain 10 or an appropriate number of old log files.

Rationale:

Kubernetes automatically rotates the log files. Retaining old log files ensures that you would have sufficient log data available for carrying out any investigation or correlation. For example, if you have set file size of 100 MB and the number of old log files to keep as 10, you would have approximately 1 GB of log data that you could potentially use for your analysis.

Impact:

None

Audit:

OpenShift audit works at the API server level, logging all requests coming to the server. Configure via `maximumRetainedFiles`.
Run the following command:

```
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .auditConfig.maximumRetainedFiles

oc get configmap config -n openshift-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .auditConfig.maximumRetainedFiles
```

Verify that `maximumRetainedFiles` is set to `10` or as appropriate.

Remediation:

Set the `maximumRetainedFiles` parameter to `10` or as an appropriate number of files.

```
maximumRetainedFiles: 10
```

Default Value:

By default, auditing is enabled.

References:

1. https://access.redhat.com/solutions/4262201
2. https://docs.openshift.com/container-platform/4.5/nodes/nodes/nodes-nodes-audit-log.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://github.com/openshift/cluster-kube-apiserver-operator/blob/master/bindata/v4.1.0/config/defaultconfig.yaml#L165-168
6. https://github.com/openshift/cluster-authentication-operator/blob/master/bindata/oauth-apiserver/deploy.yaml
7. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
8. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
9. https://github.com/kubernetes/features/issues/22

CIS Controls:

Version 6

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)

Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

Version 7

6.4 Ensure adequate storage for logs

Ensure that all systems that store logs have adequate storage space for the logs generated.

1.2.25 Ensure that the maximumFileSizeMegabytes argument is set to 100 or as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Rotate log files on reaching 100 MB or as appropriate.

Rationale:

Kubernetes automatically rotates the log files. Retaining old log files ensures that you would have sufficient log data available for carrying out any investigation or correlation. If you have set file size of 100 MB and the number of old log files to keep as 10, you would have approximately 1 GB of log data that you could potentially use for your analysis.

Impact:

None

Audit:

OpenShift audit works at the API server level, logging all requests coming to the server. Configure via `maximumFileSizeMegabytes`.
Run the following command:

```
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .auditConfig.maximumFileSizeMegabytes

oc get configmap config -n openshift-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .auditConfig.maximumFileSizeMegabytes
```

Verify that the `maximumFileSizeMegabytes` argument is set to `100` or as appropriate.

Remediation:

Set the `audit-log-maxsize` parameter to `100` or as an appropriate number.

```
maximumFileSizeMegabytes: 100
```

Default Value:

By default, auditing is not enabled.

References:

1. https://access.redhat.com/solutions/4262201
2. https://docs.openshift.com/container-platform/4.5/nodes/nodes/nodes-nodes-audit-log.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
6. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
7. https://github.com/kubernetes/features/issues/22

CIS Controls:

Version 6

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)

Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

Version 7

6.4 Ensure adequate storage for logs

Ensure that all systems that store logs have adequate storage space for the logs generated.

1.2.26 Ensure that the --request-timeout argument is set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Set global request timeout for API server requests as appropriate.

Rationale:

Setting global request timeout allows extending the API server request timeout limit to a duration appropriate to the user's connection speed. By default, it is set to 60 seconds which might be problematic on slower connections making cluster resources inaccessible once the data volume for requests exceeds what can be transmitted in 60 seconds. But, setting this timeout limit to be too large can exhaust the API server resources making it prone to Denial-of-Service attack. Hence, it is recommended to set this limit as appropriate and change the default limit of 60 seconds only if needed.

Impact:

None

Audit:

OpenShift configures the `min-request-timeout` flag via `servingInfo.requestTimeoutSeconds`, which overrides `request-timeout` in certain scenarios and provides a more balanced timeout approach than a global request-timeout. Run the following command:

```
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .servingInfo.requestTimeoutSeconds
```

Verify that the `servingInfo.requestTimeoutSeconds` argument is set to an appropriate value.

Remediation:

TBD

Default Value:

By default, `--request-timeout` is set to 60 seconds.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
4. https://github.com/kubernetes/kubernetes/pull/51415

CIS Controls:

Version 6

14.6 Enforce Detailed Audit Logging For Sensitive Information

Enforce detailed audit logging for access to nonpublic data and special authentication for sensitive data.

Version 7

14.9 Enforce Detail Logging for Access or Changes to Sensitive Data

Enforce detailed audit logging for access to sensitive data or changes to sensitive data (utilizing tools such as File Integrity Monitoring or Security Information and Event Monitoring).

1.2.27 Ensure that the --service-account-lookup argument is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Validate service account before validating token.

Rationale:

If `--service-account-lookup` is not enabled, the `apiserver` only verifies that the authentication token is valid, and does not validate that the service account token mentioned in the request is actually present in `etcd`. This allows using a service account token even after the corresponding service account is deleted. This is an example of time of check to time of use security issue.

Impact:

None

Audit:

OpenShift denies access for any OAuth Access token that does not exist in its `etcd` data store. OpenShift does not use the `service-account-lookup` flag even when it is set.
Run the following command:

```
# For OCP 4.5
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.apiServerArguments' | grep service-account-
lookup

# For OCP 4.6 and above
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq -r '.service-account-lookup'
```

For OpenShift 4.5, verify that the `service-account-lookup` argument does not exist.
For OpenShift 4.6 and above, verify that if the `--service-account-lookup` argument exists it is set to `true`.

Remediation:

TBD

Default Value:

By default, OpenShift denies access for any OAuth Access token that does not exist in its `etcd` data store. OpenShift does not use the `service-account-lookup` flag.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
3. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L145-L146
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
5. https://github.com/kubernetes/kubernetes/issues/24167
6. https://en.wikipedia.org/wiki/Time-of-check_to_time-of-use

CIS Controls:

Version 6

16 Account Monitoring and Control
Account Monitoring and Control

Version 7

16 Account Monitoring and Control
Account Monitoring and Control

1.2.28 Ensure that the --service-account-key-file argument is set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Explicitly set a service account public key file for service accounts on the apiserver.

Rationale:

By default, if no `--service-account-key-file` is specified to the `apiserver`, it uses the private key from the TLS serving certificate to verify service account tokens. To ensure that the keys for service account tokens could be rotated as needed, a separate public/private key pair should be used for signing service account tokens. Hence, the public key should be specified to the `apiserver` with `--service-account-key-file`.

Impact:

The corresponding private key must be provided to the controller manager. You would need to securely maintain the key file and rotate the keys based on your organization's key rotation policy.

Audit:

OpenShift API server does not use the service-account-key-file argument. OpenShift does not reuse the apiserver TLS key. The ServiceAccount token authenticator is configured with `serviceAccountConfig.publicKeyFiles`. OpenShift automatically manages and rotates the keys.
Run the following command:

```
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .serviceAccountPublicKeyFiles[]
```

Verify that the following is returned.
`/etc/kubernetes/static-pod-resources/configmaps/sa-token-signing-certs`
`/etc/kubernetes/static-pod-resources/configmaps/bound-sa-token-signing-certs`

Remediation:

The OpenShift API server does not use the `service-account-key-file` argument. The `ServiceAccount` token `authenticator` is configured with `serviceAccountConfig.publicKeyFiles`. OpenShift does not reuse the apiserver TLS key. This is not configurable.

Default Value:

The OpenShift API server does not use the `service-account-key-file` argument. The `ServiceAccount` token authenticator is configured with `serviceAccountConfig.publicKeyFiles`. OpenShift does not reuse the apiserver TLS key.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
4. https://github.com/kubernetes/kubernetes/issues/24167

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

1.2.29 Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

`etcd` should be configured to make use of TLS encryption for client connections.

Rationale:

`etcd` is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should be protected by client authentication. This requires the API server to identify itself to the `etcd` server using a client certificate and key.

Impact:

TLS and client certificate authentication are configured by default for `etcd`.

Audit:

OpenShift uses X.509 certificates to provide secure communication to `etcd`. OpenShift configures these automatically. OpenShift does not use the `etcd-certfile` or `etcd-keyfile` flags. Certificates are used for encrypted communication between `etcd` member peers, as well as encrypted client traffic. The following certificates are generated and used by `etcd` and other processes that communicate with `etcd`:

- Peer certificates: Used for communication between `etcd` members.
- Client certificates: Used for encrypted server-client communication. Client certificates are currently used by the API server only, and no other service should connect to `etcd` directly except for the proxy. Client secrets (`etcd-client`, `etcd-metric-client`, `etcd-metric-signer`, and `etcd-signer`) are added to the `openshift-config`, `openshift-monitoring`, and `openshift-kube-apiserver` namespaces.
- Server certificates: Used by the `etcd` server for authenticating client requests.
- Metric certificates: All metric consumers connect to proxy with `metric-client` certificates.

Run the following command:

```
# etcd Certificate File
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .storageConfig.certFile

# etcd Key File
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .storageConfig.keyFile
```

Verify that the following files exist.
`/etc/kubernetes/static-pod-resources/secrets/etcd-client/tls.crt`
`/etc/kubernetes/static-pod-resources/secrets/etcd-client/tls.key`

Remediation:

OpenShift automatically manages TLS and client certificate authentication for `etcd`. This is not configurable.

Default Value:

By default, OpenShift uses X.509 certificates to provide secure communication to `etcd`. OpenShift configures these automatically. OpenShift does not use the `etcd-certfile` or `etcd-keyfile` flags. OpenShift generates the necessary files and sets the arguments appropriately.

References:

1. https://docs.openshift.com/container-platform/4.4/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
5. https://etcd.io/

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

1.2.30 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 API server.

Rationale:

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

Impact:

TLS and client certificate authentication must be configured for your Kubernetes cluster deployment. By default, OpenShift uses X.509 certificates to provide secure connections between the API server and node/kubelet. OpenShift Container Platform monitors certificates for proper validity, for the cluster certificates it issues and manages. The OpenShift Container Platform manages certificate rotation and the alerting framework has rules to help identify when a certificate issue is about to occur.

Audit:

OpenShift uses X.509 certificates to provide secure connections between API server and node/kubelet by default. OpenShift does not use values assigned to the `tls-cert-file` or `tls-private-key-file` flags. OpenShift generates the certificate files and sets the arguments appropriately.
The API server is accessible by clients external to the cluster at `api.<cluster_name>.<base_domain>`. The administrator must set a custom default certificate to be used by the API server when serving content in order to enable clients to access the API server at a different host name or without the need to distribute the cluster-managed certificate authority (CA) certificates to the clients.
Run the following command:

```
# TLS Cert File - openshift-kube-apiserver
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .servingInfo.certFile

# TLS Key File
```

```
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .servingInfo.keyFile
```

Verify that the following files exist.
`/etc/kubernetes/static-pod-certs/secrets/service-network-serving-certkey/tls.crt`
`/etc/kubernetes/static-pod-certs/secrets/service-network-serving-certkey/tls.key`

Remediation:

OpenShift automatically manages TLS authentication for the API server communication with the node/kublet. This is not configurable.
You may optionally set a custom default certificate to be used by the API server when serving content in order to enable clients to access the API server at a different host name or without the need to distribute the cluster-managed certificate authority (CA) certificates to the clients. Follow the directions in the OpenShift documentation
User-provided certificates for the API server

Default Value:

By default, OpenShift uses X.509 certificates to provide secure connections between the API server and node/kubelet. OpenShift does not use values assigned to the `tls-cert-file` or `tls-private-key-file` flags.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#user-provided-certificates-for-the-api-server_ocp-certificates
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
5. https://rootsquash.com/2016/05/10/securing-the-kubernetes-api/
6. https://github.com/kelseyhightower/docker-kubernetes-tls-guide

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.

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

Profile Applicability:

- Level 1

Description:

Setup TLS connection on the API server.

Rationale:

API server communication contains sensitive parameters that should remain encrypted in transit. Configure the API server to serve only HTTPS traffic. If `--client-ca-file` argument is set, any request presenting a client certificate signed by one of the authorities in the `client-ca-file` is authenticated with an identity corresponding to the CommonName of the client certificate.

Impact:

TLS and client certificate authentication must be configured for your Kubernetes cluster deployment. By default, OpenShift uses X.509 certificates to provide secure connections between the API server and node/kubelet. OpenShift Container Platform monitors certificates for proper validity, for the cluster certificates it issues and manages. The OpenShift Container Platform alerting framework has rules to help identify when a certificate issue is about to occur. These rules consist of the following checks:

- API server client certificate expiration is less than five minutes.

Audit:

OpenShift uses X.509 certificates to provide secure connections between API server and node/kubelet by default. OpenShift configures the `client-ca-file` value and does not use value assigned to the `client-ca-file` flag. OpenShift generates the necessary files and sets the arguments appropriately.
The API server is accessible by clients external to the cluster at `api.<cluster_name>.<base_domain>`. The administrator must set a custom default certificate to be used by the API server when serving content in order to enable clients to access the API server at a different host name or without the need to distribute the cluster-managed certificate authority (CA) certificates to the clients.
Run the following command:

```
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .servingInfo.clientCA
```

Verify that the following file exists.
`/etc/kubernetes/static-pod-certs/configmaps/client-ca/ca-bundle.crt`

Remediation:

OpenShift automatically manages TLS authentication for the API server communication with the node/kublet. This is not configurable.
You may optionally set a custom default certificate to be used by the API server when serving content in order to enable clients to access the API server at a different host name or without the need to distribute the cluster-managed certificate authority (CA) certificates to the clients.

```
User-provided certificates must be provided in a kubernetes.io/tls type
Secret in the openshift-config namespace. Update the API server cluster
configuration, the apiserver/cluster resource, to enable the use of the user-
provided certificate.
```

Default Value:

By default, OpenShift configures the `client-ca-file` value and does not use the value assigned to the `client-ca-file` flag.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#user-provided-certificates-for-the-api-server_ocp-certificates
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
5. https://rootsquash.com/2016/05/10/securing-the-kubernetes-api/
6. https://github.com/kelseyhightower/docker-kubernetes-tls-guide

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.

1.2.32 Ensure that the --etcd-cafile argument is set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

`etcd` should be configured to make use of TLS encryption for client connections.

Rationale:

`etcd` is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should be protected by client authentication. This requires the API server to identify itself to the `etcd` server using a SSL Certificate Authority file.

Impact:

TLS and client certificate authentication must be configured for `etcd`.

Audit:

OpenShift uses X.509 certificates to provide secure communication to `etcd`. OpenShift does not use values assigned to the `etcd-cafile` argument. OpenShift generates the `etcd-cafile` and sets the arguments appropriately in the API server. OpenShift includes multiple certificate authorities (CAs) providing independent chains of trust, increasing the security posture of the cluster. The certificates generated by each CA are used to identify a particular OpenShift platform component to another OpenShift platform component. Communication with `etcd` is secured by the `etcd` serving CA.
Run the following command

```
# etcd CA File
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r .storageConfig.ca
```

Verify that the following is returned
`/etc/kubernetes/static-pod-resources/configmaps/etcd-serving-ca/ca-bundle.crt`

Remediation:

None required. OpenShift generates the `etcd-cafile` and sets the arguments appropriately in the API server. Communication with `etcd` is secured by the `etcd` serving CA.

Default Value:

By default, OpenShift uses X.509 certificates to provide secure communication to `etcd`. OpenShift does not use values assigned to `etcd-cafile`. OpenShift generates the `etcd-cafile` and sets the arguments appropriately in the API server. Communication with `etcd` is secured by the `etcd` serving CA.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#etcd-cluster-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
6. https://etcd.io/

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.

1.2.33 Ensure that the --encryption-provider-config argument is set as appropriate (Manual)

Profile Applicability:

- Level 1

Description:

Encrypt `etcd` key-value store.

Rationale:

`etcd` is a highly available key-value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should be encrypted at rest to avoid any disclosures.

Impact:

When you enable `etcd` encryption, the following OpenShift API server and Kubernetes API server resources are encrypted:

- Secrets
- ConfigMaps
- Routes
- OAuth access tokens
- OAuth authorize tokens

When you enable `etcd` encryption, encryption keys are created. These keys are rotated on a weekly basis. You must have these keys in order to restore from an `etcd` backup.

Audit:

OpenShift supports encryption of data at rest of `etcd` datastore, but it is up to the customer to configure. The `asecbc` cipher is used. Keys are stored on the filesystem of the master and automatically rotated.
Follow the steps in the documentation to encrypt the `etcd` datastore: Encrypting [etcd data | Authentication | OpenShift Container Platform 4.5]
Run the following command to review the `Encrypted` status condition for the OpenShift API server to verify that its resources were successfully encrypted:

```
# encrypt the etcd datastore
oc get openshiftapiserver -o=jsonpath='{range
```

```
.items[0].status.conditions[?(@.type=="Encrypted")]}{.reason}{"\n"}{.message}
{"\n"}'
```

The output shows `EncryptionCompleted` upon successful encryption.

- `EncryptionCompleted`
- `All resources encrypted: routes.route.openshift.io, oauthaccesstokens.oauth.openshift.io, oauthauthorizetokens.oauth.openshift.io`

If the output shows `EncryptionInProgress`, this means that encryption is still in progress. Wait a few minutes and try again.

Remediation:

Follow the OpenShift documentation for Encrypting etcd data | Authentication | OpenShift Container Platform 4.5

Default Value:

By default, `etcd` data is not encrypted in OpenShift Container Platform

References:

1. https://docs.openshift.com/container-platform/4.5/security/encrypting-etcd.html
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#etcd-cluster-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
6. https://acotten.com/post/kube17-security
7. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
8. https://github.com/kubernetes/features/issues/92

CIS Controls:

Version 6

14.5 Encrypt At Rest Sensitive Information
Sensitive information stored on systems shall be encrypted at rest and require a

secondary authentication mechanism, not integrated into the operating system, in order to access the information.

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.

1.2.34 Ensure that encryption providers are appropriately configured (Manual)

Profile Applicability:

- Level 1

Description:

Where `etcd` encryption is used, appropriate providers should be configured.

Rationale:

Where `etcd` encryption is used, it is important to ensure that the appropriate set of encryption providers is used. Currently, the `aescbc`, `kms` and `secretbox` are likely to be appropriate options.

Impact:

When you enable `etcd` encryption, the following OpenShift API server and Kubernetes API server resources are encrypted:

- Secrets
- ConfigMaps
- Routes
- OAuth access tokens
- OAuth authorize tokens

When you enable `etcd` encryption, encryption keys are created. These keys are rotated on a weekly basis. You must have these keys in order to restore from an `etcd` backup.

Audit:

OpenShift supports encryption of data at rest of `etcd` datastore, but it is up to the customer to configure. The `asecbc` cipher is used. No other ciphers are supported. Keys are stored on the filesystem of the master and automatically rotated.
Run the following command to review the Encrypted status condition for the OpenShift API server to verify that its resources were successfully encrypted:

```
# encrypt the etcd datastore
oc get openshiftapiserver -o=jsonpath='{range
.items[0].status.conditions[?(@.type=="Encrypted")]}{.reason}{"\n"}{.message}
{"\n"}'
```

The output shows EncryptionCompleted upon successful encryption.

- `EncryptionCompleted`
- `All resources encrypted: routes.route.openshift.io, oauthaccesstokens.oauth.openshift.io, oauthauthorizetokens.oauth.openshift.io`

If the output shows `EncryptionInProgress`, this means that encryption is still in progress. Wait a few minutes and try again.

Remediation:

Follow the OpenShift documentation for Encrypting etcd data | Authentication | OpenShift Container Platform 4.5

Default Value:

By default, no encryption provider is set.

References:

1. https://docs.openshift.com/container-platform/4.5/security/encrypting-etcd.html
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#etcd-cluster-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
6. https://acotten.com/post/kube17-security
7. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-apiserver/
8. https://github.com/kubernetes/features/issues/92
9. https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/#providers

CIS Controls:

Version 6

14.5 Encrypt At Rest Sensitive Information

Sensitive information stored on systems shall be encrypted at rest and require a secondary authentication mechanism, not integrated into the operating system, in order to access the information.

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.

1.2.35 Ensure that the API Server only makes use of Strong Cryptographic Ciphers (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that the API server is configured to only use strong cryptographic ciphers.

Rationale:

TLS ciphers have had a number of known vulnerabilities and weaknesses, which can reduce the protection provided by them. By default Kubernetes supports a number of TLS ciphersuites including some that have security concerns, weakening the protection provided.

Impact:

API server clients that cannot support the custom cryptographic ciphers will not be able to make connections to the API server.

Audit:

Ciphers for the API servers, authentication and the ingress controller can be configured using the `tlsSecurityProfile` parameter as of OpenShfit 4.3. The ingress controller provides external access to the API server. There are four TLS security profile types:

- Old
- Intermediate
- Modern
- Custom

Only the Old, Intermediate and Custom profiles are supported at this time. Custom provides the ability to specify individual TLS security profile parameters. Follow the steps in the documentation to configure the cipher suite for Ingress and the API server. Ingress controller configuration parameters
Run the following commands to verify the cipher suite and `minTLSversion` for the ingress operator, authentication operator, `cliconfig`, OpenShift `APIserver` and `Kube APIserver`.

```
# verify cipher suites
oc get cm -n openshift-authentication v4-0-config-system-cliconfig -o
jsonpath='{.data.v4\-0\-config\-system\-cliconfig}' | jq .servingInfo
```

```
oc get kubeapiservers.operator.openshift.io cluster -o json |jq
.spec.observedConfig.servingInfo
oc get openshiftapiservers.operator.openshift.io cluster -o json |jq
.spec.observedConfig.servingInfo
oc describe --namespace=openshift-ingress-operator ingresscontroller/default
```

Verify that the `tlsSecurityProfile` is set to the value you chose.
Note: The HAProxy Ingress controller image does not support TLS 1.3 and because the Modern profile requires TLS 1.3, it is not supported. The Ingress Operator converts the Modern profile to Intermediate. The Ingress Operator also converts the TLS 1.0 of an Old or Custom profile to 1.1, and TLS 1.3 of a Custom profile to 1.2.

Remediation:

Follow the directions above and in the OpenShift documentation Configuring Ingress.

Default Value:

By default, the TLS cipher value for the ingress controller is based on the `apiservers.config.openshift.io/cluster` resource.

References:

1. https://docs.openshift.com/container-platform/4.5/networking/ingress-operator.html#nw-ingress-controller-configuration-parameters_configuring-ingress
2. https://docs.openshift.com/container-platform/4.5/rest_api/config_apis/apiserver-config-openshift-io-v1.html
3. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-apiserver-operator_red-hat-operators
4. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#openshift-apiserver-operator_red-hat-operators
5. https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices#23-use-secure-cipher-suites

CIS Controls:

Version 6

3.4 Use Only Secure Channels For Remote System Administration

Perform all remote administration of servers, workstation, network devices, and similar equipment over secure channels. Protocols such as telnet, VNC, RDP, or others that do not actively support strong encryption should only be used if they are performed over a secondary encryption channel, such as SSL, TLS or IPSEC.

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

Use multi-factor authentication and encrypted channels for all administrative account access.

1.3 Controller Manager

This section contains recommendations relating to Controller Manager configuration flags. In OpenShift 4, the Controller Manager is managed with the cluster Controller Manager Operator. There are two operators: the OpenShift Controller operator and the Kube Controller operator. The OpenShift Controller Manager operator manages the OpenShift Controller Manager. The Kubernetes Controller Manager operator manages and updates the Kubernetes Controller Manager deployed on top of OpenShift. All calls are directed to the OpenShift Controller Manager and then Kubernetes objects are delegated to the Kubernetes Controller Manager.

1.3.1 Ensure that garbage collection is configured as appropriate (Manual)

Profile Applicability:

- Level 1

Description:

Activate garbage collector on pod termination, as appropriate.

Rationale:

Garbage collection is important to ensure sufficient resource availability and avoiding degraded performance and availability. In the worst case, the system might crash or just be unusable for a long period of time. The current setting for garbage collection is 12,500 terminated pods which might be too high for your system to sustain. Based on your system resources and tests, choose an appropriate threshold value to activate garbage collection.

Impact:

None

Audit:

Two types of garbage collection are performed on an OpenShift Container Platform node:

- Container garbage collection: Removes terminated containers.
- Image garbage collection: Removes images not referenced by any running pods.

Container garbage collection can be performed using eviction thresholds. Image garbage collection relies on disk usage as reported by cAdvisor on the node to decide which images

to remove from the node. Default values are found here https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L554-L604
The OpenShift administrator can configure how OpenShift Container Platform performs garbage collection by creating a `kubeletConfig` object for each Machine Config Pool using any combination of the following:

- soft eviction for containers
- hard eviction for containers
- eviction for images

To configure, follow the directions in https://docs.openshift.com/container-platform/4.5/nodes/nodes/nodes-nodes-garbage-collection.html#nodes-nodes-garbage-collection-configuring_nodes-nodes-configuring
To verify settings, run the following command for each updated `configpool`

```
oc get machineconfigpool

# For each machineconfigpool
oc describe machineconfigpool <name>

#For example
oc describe machineconfigpool master
oc describe machineconfigpool worker
```

Verify the values for the following are set as appropriate.
`eviction-soft`
`evictionSoftGracePeriod`
`evictionHard`
`evictionPressureTransitionPeriod`

Remediation:

To configure, follow the directions in Configuring garbage collection for containers and images

Default Value:

Container garbage collection is enabled by default and happens automatically in response to eviction thresholds being reached. Default values are found here https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L554-L604

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/nodes/nodes/nodes-nodes-garbage-collection.html
4. https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L554-L604
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/
6. https://github.com/kubernetes/kubernetes/issues/28484

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.3.2 Ensure that controller manager healthz endpoints are protected by RBAC (Automated)

Profile Applicability:

- Level 1

Description:

Disable profiling, if not needed.

Rationale:

Profiling allows for the identification of specific performance bottlenecks. It generates a significant amount of program data that could potentially be exploited to uncover system and program details. If you are not experiencing any bottlenecks and do not need the profiler for troubleshooting purposes, it is recommended to turn it off to reduce the potential attack surface.

Impact:

Profiling information would not be available.

Audit:

By default, the Controller Manager operator exposes metrics via the metrics service. Profiling data is sent to `healthzPort`, the port of the localhost `healthz` endpoint. Changing this value may disrupt components that monitor the kubelet health. To ensure the collected data is not exploited, profiling endpoints are secured via RBAC (see cluster-debugger role). By default, the profiling endpoints are accessible only by users bound to cluster-admin or cluster-debugger role.
Profiling can not be disabled.
To verify the configuration, run the following command:
Run the following command:

```
# Verify configuration for ports, livenessProbe, readinessProbe, healthz
oc -n openshift-kube-controller-manager get cm kube-controller-manager-pod -o
json | jq -r '.data."pod.yaml"' | jq '.spec.containers'

# Verify endpoints
oc -n openshift-kube-controller-manager describe endpoints

# Test to validate RBAC enabled on the controller endpoint; check with non-
admin role
oc project openshift-kube-controller-manager
```

```
POD=$(oc get pods -n openshift-kube-controller-manager -l app=kube-
controller-manager -o jsonpath='{.items[0].metadata.name}')

PORT=$(oc get pods -n openshift-kube-controller-manager -l app=kube-
controller-manager -o
jsonpath='{.items[0].spec.containers[0].ports[0].hostPort}')

# Following should return 403 Forbidden
oc rsh -n openshift-kube-controller-manager ${POD} curl
https://localhost:${PORT}/metrics -k

# Create a service account to test RBAC
oc create -n openshift-kube-controller-manager sa permission-test-sa

# Should return 403 Forbidden
SA_TOKEN=$(oc sa -n openshift-kube-controller-manager get-token permission-
test-sa)
oc rsh -n openshift-kube-controller-manager ${POD} curl
https://localhost:${PORT}/metrics -H "Authorization: Bearer $SA_TOKEN" -k

# Cleanup
oc delete -n openshift-kube-controller-manager sa permission-test-sa

# As cluster admin, should succeed
CLUSTER_ADMIN_TOKEN=$(oc whoami -t)
oc rsh -n openshift-kube-controller-manager ${POD} curl
https://localhost:${PORT}/metrics -H "Authorization: Bearer
$CLUSTER_ADMIN_TOKEN" -k
```

Verify that the livenessProbe and readinessProbe are set to path: `healthz`.
Verify that regular users cannot learn anything about the controller manager.
Verify that users with the cluster_admin role can retrieve metrics from the endpoint.

Remediation:

None required; profiling is protected by RBAC.

Default Value:

By default, the operator exposes metrics via metrics service. The metrics are collected from the OpenShift Controller Manager and the Kubernetes Controller Manager and protected by RBAC. Default profiling values are found here https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L262-L280

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators

3. https://github.com/openshift/cluster-kube-controller-manager-operator/tree/master
4. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/bootstrap-manifests/kube-controller-manager-pod.yaml
5. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/manifests/00_openshift-kube-controller-manager-ns.yaml
6. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/v4.1.0/kube-controller-manager/kubeconfig-cm.yaml
7. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/
8. https://github.com/kubernetes/community/blob/master/contributors/devel/sig-scalability/profiling.md

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.3.3 Ensure that the --use-service-account-credentials argument is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Use individual service account credentials for each controller.

Rationale:

The controller manager creates a service account per controller in the `kube-system` namespace, generates a credential for it, and builds a dedicated API client with that service account credential for each controller loop to use. Setting the `--use-service-account-credentials` to true runs each control loop within the controller manager using a separate service account credential. When used in combination with RBAC, this ensures that the control loops run with the minimum permissions required to perform their intended tasks.

Impact:

Whatever authorizer is configured for the cluster, it must grant sufficient permissions to the service accounts to perform their intended tasks. When using the RBAC authorizer, those roles are created and bound to the appropriate service accounts in the `kube-system` namespace automatically with default roles and `rolebindings` that are auto-reconciled on startup.

If using other authorization methods (ABAC, Webhook, etc), the cluster deployer is responsible for granting appropriate permissions to the service accounts (the required permissions can be seen by inspecting the `controller-roles.yaml` and `controller-role-bindings.yaml` files for the RBAC roles.

Audit:

In OpenShift, `--use-service-account-credentials` is set to `true` by default for the Controller Manager. The bootstrap configuration and overrides are available here:
kube-controller-manager-pod
bootstrap-config-overrides
Run the following command on the master node:

```
oc get configmaps config -n openshift-kube-controller-manager -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r '.extendedArguments["use-service-account-credentials"][]'
```

Verify that the `--use-service-account-credentials` argument is set to `true`.

Remediation:

The OpenShift Controller Manager operator manages and updates the OpenShift Controller Manager. The Kubernetes Controller Manager operator manages and updates the Kubernetes Controller Manager deployed on top of OpenShift. This operator is configured via KubeControllerManager custom resource.

Default Value:

By default, in OpenShift 4 `--use-service-account-credentials` is set to `true`.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
3. https://github.com/openshift/cluster-kube-controller-manager-operator/tree/master
4. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/bootstrap-manifests/kube-controller-manager-pod.yaml
5. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/config/bootstrap-config-overrides.yaml
6. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/v4.1.0/kube-controller-manager/kubeconfig-cm.yaml
7. https://github.com/openshift/cluster-openshift-controller-manager-operator/blob/release-4.5/bindata/v3.11.0/openshift-controller-manager/ds.yaml
8. https://github.com/openshift/cluster-openshift-controller-manager-operator/blob/release-4.5/bindata/v3.11.0/openshift-controller-manager/sa.yaml
9. https://github.com/openshift/cluster-openshift-controller-manager-operator/blob/release-4.5/bindata/v3.11.0/openshift-controller-manager/separate-sa-role.yaml
10. https://github.com/openshift/cluster-openshift-controller-manager-operator/blob/release-4.5/bindata/v3.11.0/openshift-controller-manager/separate-sa-rolebinding.yaml
11. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/

12. https://kubernetes.io/docs/reference/access-authn-authz/service-accounts-admin/
13. https://github.com/kubernetes/kubernetes/blob/release-1.6/plugin/pkg/auth/authorizer/rbac/bootstrappolicy/testdata/controller-roles.yaml
14. https://github.com/kubernetes/kubernetes/blob/release-1.6/plugin/pkg/auth/authorizer/rbac/bootstrappolicy/testdata/controller-role-bindings.yaml
15. https://kubernetes.io/docs/reference/access-authn-authz/rbac/#controller-roles

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.3.4 Ensure that the --service-account-private-key-file argument is set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Explicitly set a service account private key file for service accounts on the controller manager.

Rationale:

To ensure that keys for service account tokens can be rotated as needed, a separate public/private key pair should be used for signing service account tokens. The private key should be specified to the controller manager with `--service-account-private-key-file` as appropriate.

Impact:

You would need to securely maintain the key file and rotate the keys based on your organization's key rotation policy.

Audit:

OpenShift starts the Kubernetes Controller Manager with service-account-private-key-file set to `/etc/kubernetes/static-pod-resources/secrets/service-account-private-key/service-account.key`.
The bootstrap configuration and overrides are available here:
kube-controller-manager-pod
bootstrap-config-overrides
Run the following command:

```
oc get configmaps config -n openshift-kube-controller-manager -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r '.extendedArguments["service-account-private-key-file"][]'
```

Verify that the following is returned
`/etc/kubernetes/static-pod-resources/secrets/service-account-private-key/service-account.key`

Remediation:

None required. OpenShift manages the service account credentials for the scheduler automatically.

Default Value:

By default, OpenShift starts the controller manager with `service-account-private-key-file` set to `/etc/kubernetes/static-pod-resources/secrets/service-account-private-key/service-account.key`

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html
4. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#service-ca-certificates_ocp-certificates
5. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/bootstrap-manifests/kube-controller-manager-pod.yaml
6. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/config/bootstrap-config-overrides.yaml
7. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/v4.1.0/kube-controller-manager/kubeconfig-cm.yaml
8. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.3.5 Ensure that the --root-ca-file argument is set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Allow pods to verify the API server's serving certificate before establishing connections.

Rationale:

Processes running within pods that need to contact the API server must verify the API server's serving certificate. Failing to do so could be a subject to man-in-the-middle attacks.

Providing the root certificate for the API server's serving certificate to the controller manager with the `--root-ca-file` argument allows the controller manager to inject the trusted bundle into pods so that they can verify TLS connections to the API server.

Impact:

OpenShift clusters manage and maintain certificate authorities and certificates for cluster components.

Audit:

Certificates for OpenShift platform components are automatically created and rotated by the OpenShift Container Platform.
Run the following command:

```
oc get configmaps config -n openshift-kube-controller-manager -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq -r '.extendedArguments["root-ca-file"][]'
```

Verify that the `--root-ca-file` argument exists and is set to `/etc/kubernetes/static-pod-resources/configmaps/serviceaccount-ca/ca-bundle.crt`.

Remediation:

None required. Certificates for OpenShift platform components are automatically created and rotated by the OpenShift Container Platform.

Default Value:

By default, OpenShift sets the Kubernetes Controller Manager `root-ca-file` to `/etc/kubernetes/static-pod-resources/configmaps/serviceaccount-ca/ca-bundle.crt`.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
3. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html
4. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#service-ca-certificates_ocp-certificates
5. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/bootstrap-manifests/kube-controller-manager-pod.yaml
6. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/config/bootstrap-config-overrides.yaml
7. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/v4.1.0/kube-controller-manager/kubeconfig-cm.yaml
8. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/
9. https://github.com/kubernetes/kubernetes/issues/11000

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.

1.3.6 Ensure that the RotateKubeletServerCertificate argument is set to true (Automated)

Profile Applicability:

- Level 2

Description:

Enable kubelet server certificate rotation on controller-manager.

Rationale:

`RotateKubeletServerCertificate` 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 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:

Certificates for the kubelet are automatically created and rotated by the OpenShift Container Platform. The kubelet is installed automatically on every RHCOS node. The OpenShift `kubelet-serving-CA` manages certificates for the kubelet. Kubelet certificates are automatically issued and rotated.
Run the following command:

```
oc get configmaps config -n openshift-kube-controller-manager -ojson |   jq -
r '.data["config.yaml"]' |   jq -r '.extendedArguments["feature-gates"][]'
```

Verify that `RotateKubeletServerCertificate` argument exists and is set to `true`.

Remediation:

None required. Certificates for OpenShift platform components are automatically created and rotated by the OpenShift Container Platform.

Default Value:

By default, `RotateKubeletServerCertificate` is set to `true`.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
3. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/bootstrap-manifests/kube-controller-manager-pod.yaml
4. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/bootkube/config/bootstrap-config-overrides.yaml
5. https://github.com/openshift/cluster-kube-controller-manager-operator/blob/release-4.5/bindata/v4.1.0/kube-controller-manager/kubeconfig-cm.yaml
6. https://github.com/kubernetes/kubernetes/blob/release-1.11/pkg/kubelet/apis/kubeletconfig/v1beta1/types.go
7. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-tls-bootstrapping/#approval-controller
8. https://github.com/kubernetes/features/issues/267
9. https://github.com/kubernetes/kubernetes/pull/45059
10. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/

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.

1.3.7 Ensure that the --bind-address argument is set to 127.0.0.1 (Automated)

Profile Applicability:

- Level 1

Description:

Do not bind the Controller Manager service to non-loopback insecure addresses.

Rationale:

The Controller Manager API service which runs on port 10257/TCP by default is used for health and metrics information and is available without authentication or encryption. As such it should only be bound to a localhost interface, to minimize the cluster's attack surface

Impact:

None

Audit:

The `bind-address` argument is not used. The `secure-port` argument is set to `10257`. The `insecure-port` argument is set to `0`.
Run the following command:

```
oc get configmaps config -n openshift-kube-controller-manager -ojson |   jq -
r '.data["config.yaml"]' | jq '.extendedArguments["secure-port"][]'

oc get configmaps config -n openshift-kube-controller-manager -ojson |   jq -
r '.data["config.yaml"]' | jq '.extendedArguments["port"][]'

#Following should fail with a http code 403
POD=$(oc get pods -n openshift-kube-controller-manager -l app=kube-
controller-manager -o jsonpath='{.items[0].metadata.name}')

oc rsh -n openshift-kube-controller-manager -c kube-controller-manager $POD
curl https://localhost:10257/metrics -k
```

Verify that `secure-port` is set to `10257` and that `port` is set to `0`.
Verify that attempt to access the controller manager metrics fails with a HTTP code 403.

Remediation:

None required. The OpenShift operators configure this correctly.

Default Value:

By default, the `--bind-address` argument is not present, the `secure-port` argument is set to `10257` and the `port` argument is set to `0`.

References:

1. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#cluster-openshift-controller-manager-operator_red-hat-operators
2. https://docs.openshift.com/container-platform/4.5/operators/operator-reference.html#kube-controller-manager-operator_red-hat-operators
3. https://github.com/openshift/cluster-kube-controller-manager-operator
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-controller-manager/

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.

1.4 Scheduler

This section contains recommendations relating to Scheduler configuration flags.

In OpenShift 4, the Scheduler is managed with the Kubernetes Scheduler Operator. The Kubernetes Scheduler Operator manages and updates the Kubernetes Scheduler deployed on top of OpenShift Container Platform. The operator is installed with the Cluster Version Operator (CVO). The Kubernetes Scheduler Operator contains the following components:

- Operator
- Bootstrap manifest renderer
- Installer based on static pods
- Configuration observer

By default, the Operator exposes Prometheus metrics through the metrics service

1.4.1 Ensure that the healthz endpoints for the scheduler are protected by RBAC (Automated)

Profile Applicability:

- Level 1

Description:

Disable profiling, if not needed.

Rationale:

Profiling allows for the identification of specific performance bottlenecks. It generates a significant amount of program data that could potentially be exploited to uncover system and program details. If you are not experiencing any bottlenecks and do not need the profiler for troubleshooting purposes, it is recommended to turn it off to reduce the potential attack surface.

Impact:

Profiling information would not be available.

Audit:

In OpenShift 4, The Kubernetes Scheduler operator manages and updates the Kubernetes Scheduler deployed on top of OpenShift. By default, the operator exposes metrics via

metrics service. The metrics are collected from the Kubernetes Scheduler operator. Profiling data is sent to `healthzPort`, the port of the localhost `healthz` endpoint. Changing this value may disrupt components that monitor the kubelet health. The default `healthz port` value is `10251`, and the `healthz bindAddress` is `127.0.0.1`
To ensure the collected data is not exploited, profiling endpoints are secured via RBAC (see cluster-debugger role). By default, the profiling endpoints are accessible only by users bound to `cluster-admin` or `cluster-debugger` role. Profiling can not be disabled.
To verify the configuration, run the following command:
Run the following command:

```
# check configuration for ports, livenessProbe, readinessProbe, healthz

oc -n openshift-kube-scheduler get cm kube-scheduler-pod -o json | jq -r
'.data."pod.yaml"' | jq '.spec.containers'

# Test to verify endpoints

oc -n openshift-kube-scheduler describe endpoints

Test to validate RBAC enabled on the scheduler endpoint; check with non-admin
role

oc project openshift-kube-scheduler

POD=$(oc get pods -l app=openshift-kube-scheduler -o
jsonpath='{.items[0].metadata.name}')

PORT=$(oc get pod $POD -o
jsonpath='{.spec.containers[0].livenessProbe.httpGet.port}')

# Should return 403 Forbidden
oc rsh ${POD} curl http://localhost:${PORT}/metrics -k

# Create a service account to test RBAC
oc create sa permission-test-sa

# Should return 403 Forbidden
SA_TOKEN=$(oc sa get-token permission-test-sa)

oc rsh ${POD} curl http://localhost:${PORT}/metrics -H "Authorization: Bearer
$SA_TOKEN" -k

# Cleanup
oc delete sa permission-test-sa

# As cluster admin, should succeed
CLUSTER_ADMIN_TOKEN=$(oc whoami -t)
oc rsh ${POD} curl http://localhost:${PORT}/metrics -H "Authorization: Bearer
$CLUSTER_ADMIN_TOKEN" -k
```

Verify that the livenessProbe and readinessProbe are set to path: `healthz`.
Verify that only users with the `cluster_admin` role can retrieve metrics from the endpoint.

Verify that a regular user cannot get information about the scheduler. NOTE: If this check fails, please check the status of this issue:
https://bugzilla.redhat.com/show_bug.cgi?id=1889488

Remediation:

A fix to this issue: https://bugzilla.redhat.com/show_bug.cgi?id=1889488
None required. Profiling is protected by RBAC and cannot be disabled.

Default Value:

By default, profiling is enabled.

References:

1. https://github.com/openshift/cluster-kube-scheduler-operator
2. https://github.com/openshift/cluster-kube-scheduler-operator/blob/release-4.5/bindata/v4.1.0/kube-scheduler/svc.yaml
3. https://github.com/openshift/cluster-kube-scheduler-operator/blob/release-4.5/bindata/v4.1.0/kube-scheduler/pod.yaml
4. https://github.com/openshift/cluster-kube-scheduler-operator/blob/release-4.5/bindata/v4.1.0/kube-scheduler/pod.yaml#L32-L37
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/
6. https://github.com/kubernetes/community/blob/master/contributors/devel/profiling.md

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

1.4.2 Verify that the scheduler API service is protected by authentication and authorization (Automated)

Profile Applicability:

- Level 1

Description:

Do not bind the scheduler service to non-loopback insecure addresses.

Rationale:

The Scheduler API service which runs on port 10251/TCP by default is used for health and metrics information and is available without authentication or encryption. As such it should only be bound to a localhost interface, to minimize the cluster's attack surface

Impact:

None

Audit:

In OpenShift 4, The Kubernetes Scheduler operator manages and updates the Kubernetes Scheduler deployed on top of OpenShift. By default, the operator exposes metrics via metrics service. The metrics are collected from the Kubernetes Scheduler operator. Profiling data is sent to `healthzPort`, the port of the localhost `healthz` endpoint. Changing this value may disrupt components that monitor the kubelet health. The default `healthz port` value is `10251`, and the `healthz bindAddress` is `127.0.0.1`
To ensure the collected data is not exploited, profiling endpoints are secured via RBAC (see cluster-debugger role). By default, the profiling endpoints are accessible only by users bound to `cluster-admin` or `cluster-debugger` role. Profiling can not be disabled.
The bind-address argument is not used. Both authentication and authorization are in place.
https://github.com/openshift/cluster-kube-scheduler-operator
Run the following command:

```
# to verify endpoints

oc -n openshift-kube-scheduler describe endpoints

# To verify that bind-adress is not used in the configuration and that port
is set to 0
oc -n openshift-kube-scheduler get cm kube-scheduler-pod -o json | jq -r
'.data."pod.yaml"' | jq '.spec.containers'
```

```
# To test for RBAC:
oc project openshift-kube-scheduler

POD=$(oc get pods -l app=openshift-kube-scheduler -o
jsonpath='{.items[0].metadata.name}')

POD_IP=$(oc get pods -l app=openshift-kube-scheduler -o
jsonpath='{.items[0].status.podIP}')

PORT=$(oc get pod $POD -o
jsonpath='{.spec.containers[0].livenessProbe.httpGet.port}')

# Should return a 403
oc rsh ${POD} curl http://${POD_IP}:${PORT}/metrics

# Create a service account to test RBAC
oc create sa permission-test-sa

# Should return 403 Forbidden
SA_TOKEN=$(oc sa get-token permission-test-sa)
oc rsh ${POD} curl http://localhost:${PORT}/metrics -H "Authorization: Bearer
$SA_TOKEN" -k

# Cleanup
oc delete sa permission-test-sa

# As cluster admin, should succeed
CLUSTER_ADMIN_TOKEN=$(oc whoami -t)
oc rsh ${POD} curl http://localhost:${PORT}/metrics -H "Authorization: Bearer
$CLUSTER_ADMIN_TOKEN" -k
```

Verify that the --bind-address argument is not present and that `healthz` is bound to `port 10251`.
Verify that only users with the cluster_admin role can retrieve metrics from the endpoint.
Verify that a regular user cannot get information about the scheduler. NOTE: If this check fails, please check the status of this issue:
https://bugzilla.redhat.com/show_bug.cgi?id=1889488

Remediation:

By default, the `--bind-address` argument is not present, the readinessProbe and `livenessProbe` arguments are set to `10251` and the `port` argument is set to `0`.
Check the status of this issue: https://bugzilla.redhat.com/show_bug.cgi?id=1889488

Default Value:

By default, the `--bind-address parameter` is not used.

References:

1. https://github.com/openshift/cluster-kube-scheduler-operator

2. https://github.com/openshift/cluster-kube-scheduler-operator/blob/release-4.5/bindata/v4.1.0/kube-scheduler/svc.yaml
3. https://github.com/openshift/cluster-kube-scheduler-operator/blob/release-4.5/bindata/v4.1.0/kube-scheduler/pod.yaml
4. https://github.com/openshift/cluster-kube-scheduler-operator/blob/release-4.5/bindata/v4.1.0/kube-scheduler/pod.yaml#L32-L37
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/

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.

2 etcd

This section covers recommendations for etcd configuration. The OpenShift cluster-etcd-operator (CEO) is an operator that handles the scaling of etcd during cluster bootstrap and regular operation. The operator also manages provisioning etcd dependencies such as TLS certificates.

OpenShift uses X.509 certificates to provide secure communication to etcd. OpenShift generates these files and sets the arguments appropriately. etcd certificates are used for encrypted communication between etcd member peers, as well as encrypted client traffic. The following certificates are generated and used by etcd and other processes that communicate with etcd:

- Peer certificates: Used for communication between etcd members.
- Client certificates: Used for encrypted server-client communication. Client certificates are currently used by the API server only, and no other service should connect to etcd directly except for the proxy. Client secrets (etcd-client, etcd-metric-client, etcd-metric-signer, and etcd-signer) are added to the openshift-config, openshift-monitoring, and openshift-kube-apiserver namespaces.
- Server certificates: Used by the etcd server for authenticating client requests.
- Metric certificates: All metric consumers connect to proxy with metric-client certificates.

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

Profile Applicability:

- Level 1

Description:

Configure TLS encryption for the `etcd` service.

Rationale:

`etcd` is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should be encrypted in transit.

Impact:

Client connections only over TLS would be served.

Audit:

OpenShift uses X.509 certificates to provide secure communication to `etcd`. OpenShift generates these files and sets the arguments appropriately. OpenShift does not use the `etcd-certfile` or `etcd-keyfile` flags.
Keys and certificates for control plane components like `kube-apiserver`, `kube-controller-manager`, `kube-scheduler` and `etcd` are stored with their respective static pod configurations in the directory `/etc/kubernetes/static-pod-resources/*/secrets`.
Run the following command:

```
# needs verification

# For --cert-file
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | sed 's/.*\(--cert-file=[^ ]*\).*/\1/'
done

# For --key-file
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | sed 's/.*\(--key-file=[^ ]*\).*/\1/'
done
```

Verify that cert-file and key-file values are returned for each etcd member.
`--cert-file=/etc/kubernetes/static-pod-certs/secrets/etcd-all-serving/etcd-serving-${ETCD_DNS_NAME}.crt`
`--key-file=/etc/kubernetes/static-pod-certs/secrets/etcd-all-serving/etcd-serving-${ETCD_DNS_NAME}.key`
For example:
`--cert-file=/etc/kubernetes/static-pod-certs/secrets/etcd-all-serving/etcd-serving-ip-10-0-165-75.us-east-2.compute.internal.crt`
`--key-file=/etc/kubernetes/static-pod-certs/secrets/etcd-all-serving/etcd-serving-ip-10-0-165-75.us-east-2.compute.internal.key`

Remediation:

OpenShift does not use the `etcd-certfile` or `etcd-keyfile` flags. Certificates for `etcd` are managed by the `etcd` cluster operator.

Default Value:

By default, `etcd` communication is secured with X.509 certificates.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://github.com/openshift/cluster-etcd-operator
3. https://github.com/openshift/cluster-etcd-operator/blob/master/bindata/etcd/pod.yaml#L154-L167
4. https://etcd.io/
5. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/

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.

2.2 Ensure that the --client-cert-auth argument is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Enable client authentication on etcd service.

Rationale:

etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should not be available to unauthenticated clients. You should enable the client authentication via valid certificates to secure the access to the etcd service.

Impact:

All clients attempting to access the etcd server will require a valid client certificate.

Audit:

OpenShift uses X.509 certificates to provide secure communication to etcd. OpenShift installation generates these files and sets the arguments appropriately. The following certificates are generated and used by etcd and other processes that communicate with etcd:

- Client certificates: Client certificates are currently used by the API server only, and no other service should connect to etcd directly except for the proxy. Client secrets (`etcd-client`, `etcd-metric-client`, `etcd-metric-signer`, and `etcd-signer`) are added to the `openshift-config`, `openshift-monitoring`, and `openshift-kube-apiserver` namespaces.
- Server certificates: Used by the etcd server for authenticating client requests.

Run the following command on the etcd server node:

```
# needs verification
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | sed 's/.*\(--client-cert-auth=[^ ]*\).*/\1/'
done
```

Verify that the `--client-cert-auth` argument is set to `true` for each etcd member.

Remediation:

This setting is managed by the cluster etcd operator. No remediation required.

Default Value:

By default, `client-cert-auth` is set to `true`.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://github.com/openshift/cluster-etcd-operator
3. https://github.com/openshift/cluster-etcd-operator/blob/release-4.5/bindata/etcd/pod.yaml#L154-L167
4. https://github.com/openshift/cluster-etcd-operator/blob/master/bindata/etcd/pod.yaml#L154-L167
5. https://etcd.io/
6. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/
7. https://etcd.io/#client-cert-auth

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

2.3 Ensure that the --auto-tls argument is not set to true (Automated)

Profile Applicability:

- Level 1

Description:

Do not use self-signed certificates for TLS.

Rationale:

etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should not be available to unauthenticated clients. You should enable the client authentication via valid certificates to secure the access to the etcd service.

Impact:

Clients will not be able to use self-signed certificates for TLS.

Audit:

OpenShift configures etcd with secure communication. Openshift installs etcd as static pods on control plane nodes, and mounts the configuration files from /etc/etcd/ on the host. The `etcd.conf` file includes `auto-tls` configurations as referenced in `/etc/etcd/etcd.conf`.
OpenShift 4 includes multiple CAs providing independent chains of trust, which ensure that a platform CA will never accidentally sign a certificate that can be used for the wrong purpose, increasing the security posture of the cluster.
These internal self-signing CAs enable automation because the key is known to the cluster. The certificates generated by each CA are used to identify a particular OpenShift platform component to another OpenShift platform component. The OpenShift CAs are managed by the cluster and are only used within the cluster.

- Each cluster CA can only issue certificates for its own purpose within its own cluster.
- CAs for one OpenShift cluster cannot influence CAs for a different OpenShift cluster, thus avoiding cross-cluster interference.
- Cluster CAs cannot be influenced by an external CA that the cluster does not control.

Run the following command:

```
# needs verification
```

```
# Returns 0 if found, 1 if not found
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | grep -- --auto-tls=true 2>&1>/dev/null ; \
      echo $?
done
```

Verify that 1 is returned for each etcd member.

Remediation:

This setting is managed by the cluster etcd operator. No remediation required.

Default Value:

By default, OpenShift configures etcd to use a cluster CA which creates self-signed certificates. These internal self-signing CAs enable automation because the key is known to the cluster. The certificates generated by each CA are used to identify a particular OpenShift platform component to another OpenShift platform component. The OpenShift CAs are managed by the cluster and are only used within the cluster.

- Each cluster CA can only issue certificates for its own purpose within its own cluster.
- CAs for one OpenShift cluster cannot influence CAs for a different OpenShift cluster, thus avoiding cross-cluster interference.
- Cluster CAs cannot be influenced by an external CA that the cluster does not control.

This configuration cannot be changed.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://github.com/openshift/cluster-etcd-operator
3. https://github.com/openshift/cluster-etcd-operator/blob/release-4.5/bindata/etcd/pod.yaml#L154-L167
4. https://github.com/openshift/cluster-etcd-operator/blob/master/bindata/etcd/pod.yaml#L154-L167
5. https://etcd.io/
6. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/
7. https://etcd.io/#auto-tls

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.

2.4 Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

etcd should be configured to make use of TLS encryption for peer connections.

Rationale:

etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should be encrypted in transit and also amongst peers in the etcd clusters.

Impact:

etcd cluster peers are set up TLS for their communication.

Audit:

OpenShift uses X.509 certificates to provide secure communication to etcd. OpenShift generates these files and sets the arguments appropriately. etcd certificates are used for encrypted communication between etcd member peers, as well as encrypted client traffic. Peer certificates are generated and used for communication between etcd members. Openshift installs etcd as static pods on control plane nodes, and mounts the configuration files from `/etc/etcd/` on the host. The `etcd.conf` file includes `peer-cert-file` and `peer-key-file` configurations as referenced in `/etc/etcd/etcd.conf`.
Run the following command:

```
# needs verification

# For --peer-cert-file
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | sed 's/.*\(--peer-cert-file=[^ ]*\).*/\1/'
done

# For --peer-key-file
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
```

```
      ps -o command= -C etcd | sed 's/.*\(--peer-key-file=[^ ]*\).*/\1/'
done
```

Verify that the following is returned for each etcd member.

```
--peer-cert-file=/etc/kubernetes/static-pod-certs/secrets/etcd-all-peer/etcd-peer-${ETCD_DNS_NAME}.crt
--peer-key-file=/etc/kubernetes/static-pod-certs/secrets/etcd-all-peer/etcd-peer-${ETCD_DNS_NAME}.key
```

For example

```
--peer-cert-file=/etc/kubernetes/static-pod-certs/secrets/etcd-all-peer/etcd-peer-ip-10-0-158-52.us-east-2.compute.internal.crt
--peer-key-file=/etc/kubernetes/static-pod-certs/secrets/etcd-all-peer/etcd-peer-ip-10-0-158-52.us-east-2.compute.internal.key
```

Remediation:

None. This configuration is managed by the etcd operator.

Default Value:

By default, peer communication over TLS is configured.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://github.com/openshift/cluster-etcd-operator
3. https://github.com/openshift/cluster-etcd-operator/blob/release-4.5/bindata/etcd/pod.yaml#L154-L167
4. https://github.com/openshift/cluster-etcd-operator/blob/master/bindata/etcd/pod.yaml#L154-L167
5. https://etcd.io/
6. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/

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.

2.5 Ensure that the --peer-client-cert-auth argument is set to true (Automated)

Profile Applicability:

- Level 1

Description:

etcd should be configured for peer authentication.

Rationale:

etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should be accessible only by authenticated etcd peers in the etcd cluster.

Impact:

All peers attempting to communicate with the etcd server require a valid client certificate for authentication.

Audit:

OpenShift uses X.509 certificates to provide secure communication to etcd. OpenShift generates these files and sets the arguments appropriately. etcd certificates are used for encrypted communication between etcd member peers, as well as encrypted client traffic. Peer certificates are generated and used for communication between etcd members. Openshift installs etcd as static pods on control plane nodes, and mounts the configuration files from `/etc/etcd/` on the host. The `etcd.conf` file includes `peer-client-cert-auth` configurations as referenced in `/etc/etcd/etcd.conf`.
Run the following command:

```
# needs verification
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | sed 's/.*\(--peer-client-cert-auth=[^
]*\).*/\1/'
done
```

Verify that the `--peer-client-cert-auth` argument is set to `true` for each etcd member.

Remediation:

This setting is managed by the cluster etcd operator. No remediation required.

Default Value:

By default, `--peer-client-cert-auth` argument is set to `true`.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://github.com/openshift/cluster-etcd-operator
3. https://github.com/openshift/cluster-etcd-operator/blob/release-4.5/bindata/etcd/pod.yaml#L154-L167
4. https://github.com/openshift/cluster-etcd-operator/blob/release-4.5/bindata/etcd/pod.yaml#L154-L167
5. https://etcd.io/
6. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/
7. https://etcd.io/#peer-client-cert-auth

CIS Controls:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected 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.

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.

2.6 Ensure that the --peer-auto-tls argument is not set to true (Automated)

Profile Applicability:

- Level 1

Description:

Do not use automatically generated self-signed certificates for TLS connections between peers.

Rationale:

etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should be accessible only by authenticated etcd peers in the etcd cluster. Hence, do not use self-signed certificates for authentication.

Impact:

All peers attempting to communicate with the etcd server require a valid client certificate for authentication.

Audit:

OpenShift does not use the `--peer-auto-tls` argument. OpenShift 4 includes multiple CAs providing independent chains of trust, which ensure that a platform CA will never accidentally sign a certificate that can be used for the wrong purpose, increasing the security posture of the cluster.
These internal self-signing CAs enable automation because the key is known to the cluster. The certificates generated by each CA are used to identify a particular OpenShift platform component to another OpenShift platform component. The OpenShift CAs are managed by the cluster and are only used within the cluster. This means that

- Each cluster CA can only issue certificates for its own purpose within its own cluster.
- CAs for one OpenShift cluster cannot influence CAs for a different OpenShift cluster, thus avoiding cross-cluster interference.

Cluster CAs cannot be influenced by an external CA that the cluster does not control.
Run the following command:

```
# needs verification

# Returns 0 if found, 1 if not found
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | grep -- --peer-auto-tls=true 2>&1>/dev/null ;
\
      echo $?
done
```

Verify that 1 is returned for each etcd member.

Remediation:

This setting is managed by the cluster etcd operator. No remediation required.

Default Value:

OpenShift does not use the `--peer-auto-tls` argument. By default, OpenShift configures etcd to use a cluster CA which creates self-signed certificates. These internal self-signing CAs enable automation because the key is known to the cluster. The certificates generated by each CA are used to identify a particular OpenShift platform component to another OpenShift platform component. The OpenShift CAs are managed by the cluster and are only used within the cluster. This means that

- Each cluster CA can only issue certificates for its own purpose within its own cluster.
- CAs for one OpenShift cluster cannot influence CAs for a different OpenShift cluster, thus avoiding cross-cluster interference.
- Cluster CAs cannot be influenced by an external CA that the cluster does not control.

This configuration cannot be changed.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://github.com/openshift/cluster-etcd-operator
3. https://github.com/openshift/cluster-etcd-operator/blob/release-4.5/bindata/etcd/pod.yaml#L154-L167
4. https://github.com/openshift/cluster-etcd-operator/blob/master/bindata/etcd/pod.yaml#L154-L167
5. https://etcd.io/
6. https://etcd.io/#peer-auto-tls
7. https://etcd.io/#peer-auto-tls

CIS Controls:

Version 6

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

Version 7

4 Controlled Use of Administrative Privileges
Controlled Use of Administrative Privileges

2.7 Ensure that a unique Certificate Authority is used for etcd (Manual)

Profile Applicability:

- Level 2

Description:

Use a different certificate authority for etcd from the one used for Kubernetes.

Rationale:

etcd is a highly available key-value store used by Kubernetes deployments for persistent storage of all of its REST API objects. Its access should be restricted to specifically designated clients and peers only.

Authentication to etcd is based on whether the certificate presented was issued by a trusted certificate authority. There is no checking of certificate attributes such as common name or subject alternative name. As such, if any attackers were able to gain access to any certificate issued by the trusted certificate authority, they would be able to gain full access to the etcd database.

Impact:

Additional management of the certificates and keys for the dedicated certificate authority will be required.

Audit:

OpenShift 4 includes multiple CAs providing independent chains of trust, which ensure that a platform CA will never accidentally sign a certificate that can be used for the wrong purpose, increasing the security posture of the cluster. OpenShift uses a separate CA for etcd.
These internal self-signing CAs enable automation because the key is known to the cluster. The certificates generated by each CA are used to identify a particular OpenShift platform component to another OpenShift platform component. The OpenShift CAs are managed by the cluster and are only used within the cluster. This means that

- Each cluster CA can only issue certificates for its own purpose within its own cluster.
- CAs for one OpenShift cluster cannot influence CAs for a different OpenShift cluster, thus avoiding cross-cluster interference.

Cluster CAs cannot be influenced by an external CA that the cluster does not control. Run the following command:

```
# needs verification
for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | sed 's/.*\(--trusted-ca-file=[^ ]*\).*/\1/'
done

for i in $(oc get pods -oname -n openshift-etcd)
do
    oc exec -n openshift-etcd -c etcd $i -- \
      ps -o command= -C etcd | sed 's/.*\(--peer-trusted-ca-file=[^
]*\).*/\1/'
done
```

Verify that `--trusted-ca-file=/etc/kubernetes/static-pod-certs/configmaps/etcd-serving-ca/ca-bundle.crt` and `--peer-trusted-ca-file=/etc/kubernetes/static-pod-certs/configmaps/etcd-peer-client-ca/ca-bundle.crt` are returned for each member.

Remediation:

None required. Certificates for etcd are managed by the OpenShift cluster etcd operator.

Default Value:

By default, in OpenShift 4, communication with etcd is secured by the etcd serving CA.

References:

1. https://docs.openshift.com/container-platform/4.5/security/certificate-types-descriptions.html#etcd-certificates_ocp-certificates
2. https://github.com/openshift/cluster-etcd-operator
3. https://github.com/openshift/cluster-etcd-operator/blob/release-4.5/bindata/etcd/pod.yaml#L154-L167
4. https://github.com/openshift/cluster-etcd-operator/blob/master/bindata/etcd/pod.yaml#L154-L167
5. https://etcd.io/

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 Control Plane Configuration

This section contains recommendations for cluster-wide areas, such as authentication and logging. Unlike section 1 these recommendations should apply to all deployments.

3.1 Authentication and Authorization

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

Profile Applicability:

- Level 2

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:

For users to interact with OpenShift Container Platform, they must first authenticate to the cluster. The authentication layer identifies the user with requests to the OpenShift Container Platform API. The authorization layer then uses information about the requesting user to determine if the request is allowed. Understanding authentication | Authentication | OpenShift Container Platform 4.5
The OpenShift Container Platform includes a built-in OAuth server for token-based authentication. Developers and administrators obtain OAuth access tokens to authenticate themselves to the API. It is recommended for an administrator to configure OAuth to specify an identity provider after the cluster is installed. User access to the cluster is managed through the identity provider. Understanding identity provider configuration |

Authentication | OpenShift Container Platform 4.5
Run the following commands:

```
# needs verification

# To verify user authentication is enabled
oc describe authentication

# To verify that an identity provider is configured
oc get identity

# To verify that a custom cluster-admin user exists
oc get clusterrolebindings -o=custom-
columns=NAME:.metadata.name,ROLE:.roleRef.name,SUBJECT:.subjects[*].kind |
grep cluster-admin | grep User

# To verity that kbueadmin is removed, no results should be returned
oc get secrets kubeadmin -n kube-system
```

Verify that authentication is running.
Verify that at least one identity provider is configured.
Verify that at least one user has cluster-admin role. For example
`NAME: cluster-admin-0 ROLE: cluster-admin SUBJECT:*.kind User`
Verify that the `kubeadmin` user no longer exists

Remediation:

Configure an identity provider for the OpenShift cluster. Understanding identity provider configuration | Authentication | OpenShift Container Platform 4.5. Once an identity provider has been defined, you can use RBAC to define and apply permissions. After you define an identity provider and create a new `cluster-admin` user, remove the `kubeadmin` user to improve cluster security.

Default Value:

By default, only a `kubeadmin` user exists on your cluster. To specify an identity provider, you must create a Custom Resource (CR) that describes that identity provider and add it to the cluster.

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/understanding-identity-provider.html
2. https://docs.openshift.com/container-platform/4.5/authentication/using-rbac.html#authorization-overview_using-rbac
3. https://docs.openshift.com/container-platform/4.5/authentication/remove-kubeadmin.html

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.

3.2 Logging

3.2.1 Ensure that a minimal audit policy is created (Automated)

Profile Applicability:

- Level 1

Description:

Kubernetes can audit the details of requests made to the API server. The `--audit-policy-file` flag must be set for this logging to be enabled.

Rationale:

Logging is an important detective control for all systems, to detect potential unauthorised access.

Impact:

Audit logs will be created on the master nodes, which will consume disk space. Care should be taken to avoid generating too large volumes of log information as this could impact the available of the cluster nodes.

Audit:

In OpenShift, auditing of the API Server is on by default. Audit provides a security-relevant chronological set of records documenting the sequence of activities that have affected the system by individual users, administrators, or other components of the system. Audit works at the API server level, logging all requests coming to the server. Each audit log contains two entries:
The request line containing:
A Unique ID allowing to match the response line (see #2)

- The source IP of the request
- The HTTP method being invoked
- The original user invoking the operation
- The impersonated user for the operation (self meaning himself)
- The impersonated group for the operation (lookup meaning user's group)
- The namespace of the request or
- The URI as requested

The response line containing:

- The unique ID from #1
- The response code

You can view logs for the OpenShift Container Platform API server or the Kubernetes API server for each master node. Follow the steps in documentation. Viewing the audit log

```
# needs verification

#To view kube apiserver log files
oc adm node-logs --role=master --path=kube-apiserver/

#To view openshift apiserver log files
oc adm node-logs --role=master --path=openshift-apiserver/

#To verify kube apiserver audit config
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq '.auditConfig[]'

#To verify openshift  apiserver audit config
oc get configmap config -n openshift-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq '.auditConfig[]'
```

Verify that log files are returned.
Verify the audit log configuration.

Remediation:

No remediation required.

Default Value:

By default, in OpenShift, auditing of the API Server is on.

References:

1. https://docs.openshift.com/container-platform/4.5/nodes/nodes/nodes-nodes-audit-log.html
2. https://github.com/openshift/cluster-kube-apiserver-operator/blob/master/bindata/v4.1.0/config/defaultconfig.yaml#L17-L31
3. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/

CIS Controls:

Version 7

6.2 Activate audit logging
Ensure that local logging has been enabled on all systems and networking devices.

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

Profile Applicability:

- 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 an effective part of a security environment.

Impact:

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

Audit:

Review the audit policy provided for the cluster and ensure that it covers at least the following areas:

- Access to Secrets managed by the cluster. Care should be taken to only log Metadata for requests to Secrets, ConfigMaps, and TokenReviews, in order to avoid the risk of logging sensitive data.
- Modification of pod and deployment objects.
- Use of `pods/exec`, `pods/portforward`, `pods/proxy and services/proxy`.

For most requests, minimally logging at the Metadata level is recommended (the most basic level of logging).
Audit policy is supported as of OpenShift 4.6, but not in earlier versions. You can configure the audit feature to set log level, retention policy, and the type of events to log. You can set the log level settings for an overall component or the API server to one of the following. The setting can be different for each setting.

```
# needs verification

#To verify openshift apiserver audit config
oc get configmap config -n openshift-kube-apiserver -ojson | \
  jq -r '.data["config.yaml"]' | \
  jq '.auditConfig.policyConfiguration.rules[]'

#To verify kube apiserver audit config
oc get configmap config -n openshift-apiserver -ojson | \
```

```
jq -r '.data["config.yaml"]' | \
jq '.auditConfig.policyConfiguration.rules[]'
```

Remediation:

In OpenShift 4.6 and higher, if appropriate for your needs, modify the audit policy.

Default Value:

By default, OpenShift 4 logs audit data for the API server. In OpenShift 4.6 and above, the audit policy can be configured.

References:

1. https://docs.openshift.com/container-platform/4.6/nodes/nodes/nodes-nodes-audit-config.html
2. https://docs.openshift.com/container-platform/4.5/nodes/nodes/nodes-nodes-audit-log.html#nodes-pods-audit-log-basic_nodes-nodes-audit-log
3. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L47-L77
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L34-L78
5. https://github.com/k8scop/k8s-security-dashboard/blob/master/configs/kubernetes/adv-audit.yaml
6. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/#audit-policy
7. https://github.com/falcosecurity/falco/blob/master/examples/k8s_audit_config/audit-policy.yaml
8. https://github.com/kubernetes/kubernetes/blob/master/cluster/gce/gci/configure-helper.sh#L735

CIS Controls:

Version 6

14.6 Enforce Detailed Audit Logging For Sensitive Information

Enforce detailed audit logging for access to nonpublic data and special authentication for sensitive data.

Version 7

14.9 Enforce Detail Logging for Access or Changes to Sensitive Data

Enforce detailed audit logging for access to sensitive data or changes to sensitive data (utilizing tools such as File Integrity Monitoring or Security Information and Event Monitoring).

4 Worker Nodes

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

Note that these components may also run on Kubernetes master nodes, so the recommendations in this section should be applied to master nodes as well as worker nodes where the master nodes make use of these components.

4.1 Worker Node Configuration Files

This section covers recommendations for configuration files on the worker nodes. As the same files exist on the master nodes, the same commands should be run on all nodes.

In OpenShift 4, node configuration files are managed by the Machine Config Operator.

4.1.1 Ensure that the kubelet service file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the kubelet service file has permissions of 644 or more restrictive.

Rationale:

The kubelet service file controls various parameters that set the behavior 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.

Impact:

None

Audit:

Kubelet is run as a `systemd` unit and its configuration file is created with `644` permissions. Run the following command:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host stat -c %a
/etc/systemd/system/kubelet.service
done
```

Verify that the permissions are `644` or more restrictive.

Remediation:

By default, the kubelet service file has permissions of `644`.

Default Value:

By default, the kubelet service file has permissions of `644`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
4. https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#44-joining-your-nodes
5. https://kubernetes.io/docs/reference/setup-tools/kubeadm/#kubelet-drop-in

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.

4.1.2 Ensure that the kubelet service file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the kubelet service file ownership is set to `root:root`.

Rationale:

The kubelet service file controls various parameters that set the behavior of 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:

Run the following command:

```
# needs verification

# Should return root:root for each node
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host stat -c %U:%G
/etc/systemd/system/kubelet.service
done
```

Verify that the ownership is set to `root:root`.

Remediation:

By default, the kubelet service file has ownership of `root:root`.

Default Value:

By default, `kubelet` service file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
4. https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#44-joining-your-nodes
5. https://kubernetes.io/docs/reference/setup-tools/kubeadm/#kubelet-drop-in

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.

4.1.3 If proxy kubeconfig file exists ensure permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

If `kube-proxy` 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 `kube-proxy` kubeconfig file controls various parameters of the `kube-proxy` 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 `kube-proxy` 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:

In OpenShift 4, the `kube-proxy` runs within the `sdn` pods, which copies the `kubeconfig` from a configmap to the container at `/tmp/kubeconfig`, with `644` permissions.
Run the following command:

```
# needs verification
for i in $(oc get pods -n openshift-sdn -l app=sdn -oname)
do
    oc exec -n openshift-sdn $i -- \
      stat -Lc %a /config/kube-proxy-config.yaml
done
```

Verify that the `kube-proxy-config.yaml` file has permissions of `644`.

Remediation:

None needed.

Default Value:

By default, `kube-proxy` config file has permissions of `644`.

References:

1. https://docs.openshift.com/container-platform/4.5/networking/openshift_sdn/configuring-kube-proxy.html
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/

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.

4.1.4 If proxy kubeconfig file exists ensure ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The kubeconfig file for `kube-proxy` controls various parameters for the `kube-proxy` 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:

In OpenShift 4, the `kube-proxy` runs within the `sdn` pods, which copies the `kubeconfig` from a configmap to the container at `/tmp/kubeconfig`, with root:root ownership.
Run the following command:

```
for i in $(oc get pods -n openshift-sdn -l app=sdn -oname)
do
    oc exec -n openshift-sdn $i -- \
      stat -Lc %U:%G /config/kube-proxy-config.yaml
done
```

Verify that the `kube-proxy-config.yaml` file has ownership root:root.

Remediation:

None required. The configuration is managed by OpenShift operators.

Default Value:

By default, `proxy` file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.5/networking/openshift_sdn/configuring-kube-proxy.html
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/

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 the --kubeconfig kubelet.conf file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `kubelet.conf` file has permissions of `644` or more restrictive.

Rationale:

The `kubelet.conf` file is the kubeconfig file for the node, and controls various parameters that set the behavior and identity of 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.

Impact:

None

Audit:

The node's `kubeconfig` is created with `644` permissions.
Run the following command:

```
# Check permissions
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host stat -c %a
/etc/kubernetes/kubelet.conf
done
```

Verify that the permissions are `644`.

Remediation:

None required.

Default Value:

By default, `kubelet.conf` file has permissions of `644`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/master/01-master-kubelet/_base/files/kubelet.yaml
4. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/worker/01-worker-kubelet/_base/files/kubelet.yaml
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

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.

4.1.6 Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the `kubelet.conf` file ownership is set to `root:root`.

Rationale:

The `kubelet.conf` file is the kubeconfig file for the node, and controls various parameters that set the behavior and identity of 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:

The node's `kubeconfig` is created with `root:root` ownership.
Run the following command:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host stat -c %U:%G
/etc/kubernetes/kubelet.conf
done
```

Verify that the ownership is set to `root:root`.

Remediation:

None required.

Default Value:

By default, `kubelet.conf` file ownership is set to `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/master/01-master-kubelet/_base/files/kubelet.yaml
4. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/worker/01-worker-kubelet/_base/files/kubelet.yaml
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

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.

4.1.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the certificate authorities file has permissions of `644` or more restrictive.

Rationale:

The certificate authorities file controls the authorities used to validate API requests. 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:

The Client CA location for the `kubelet` is defined in `/etc/kubernetes/kubelet.conf`. The `/etc/kubernetes/kubelet-ca.crt` file has permissions `644`.
Run the following command:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host stat -c %a
/etc/kubernetes/kubelet-ca.crt
done
```

Verify that the permissions are `644`.

Remediation:

None required.

Default Value:

By default, in OpenShift 4, the `--client-ca-file` is set to `/etc/kubernetes/kubelet-ca.crt` with permissions set to `644`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/master/01-master-kubelet/_base/files/kubelet.yaml
4. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/worker/01-worker-kubelet/_base/files/kubelet.yaml
5. https://kubernetes.io/docs/reference/access-authn-authz/authentication/#x509-client-certs

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.

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected 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.

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.

4.1.8 Ensure that the client certificate authorities file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that the certificate authorities file ownership is set to `root:root`.

Rationale:

The certificate authorities file controls the authorities used to validate API requests. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Impact:

None

Audit:

The Client CA location for the `kubelet` is defined in `/etc/kubernetes/kubelet.conf`. The `/etc/kubernetes/kubelet-ca.crt` file has ownership `root:root`.
Run the following command:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host stat -c %U:%G
/etc/kubernetes/kubelet-ca.crt
done
```

Verify that the ownership is set to `root:root`.

Remediation:

None required.

Default Value:

By default, in OpenShift 4, the `--client-ca-file` is set to `/etc/kubernetes/kubelet-ca.crt` with ownership `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/master/01-master-kubelet/_base/files/kubelet.yaml
4. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/worker/01-worker-kubelet/_base/files/kubelet.yaml
5. https://kubernetes.io/docs/reference/access-authn-authz/authentication/#x509-client-certs

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Impact:

None

Audit:

In OpenShift 4, the `kublet.conf` file is managed by the Machine Config Operator. The kubelet config file is found at `/var/lib/kubelet/kubeconfig` with file permissions set to `600`.
Run the command:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host stat -c %a
/var/lib/kubelet/kubeconfig
done
```

Verify that the permissions are `600`.

Remediation:

None required.

Default Value:

By default, in OpenShift 4, the `/var/lib/kubelet/config.yaml` file has permissions of `600`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/docs/KubeletConfigDesign.md
4. https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Impact:

None

Audit:

In OpenShift 4, the kublet config file is managed by the Machine Config Operator. The kubelet config file is found at `/var/lib/kubelet/kubeconfig` with ownership set to `root:root`.
Run the command:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host stat -c %U:%G
/var/lib/kubelet/kubeconfig
done
```

Verify that the ownership is set to `root:root`.

Remediation:

None required.

Default Value:

By default, `/var/lib/kubelet/config.yaml` file is owned by `root:root`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/docs/KubeletConfigDesign.md
4. https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/

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 Kubelet

This section contains recommendations for kubelet configuration. In OpenShift 4, the kubelet is managed by the Machine Config Operator.

4.2.1 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:

In OpenShift 4, the kublet config file is managed by the Machine Config Operator and `anonymous-auth` is set to `false` by default.
Run the following command on each node:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host grep -B4 -A1 anonymous:
/etc/kubernetes/kubelet.conf
done
```

Verify that the `anonymous-auth` argument is set to `false`.

Remediation:

Follow the instructions in the documentation to create a Kubelet config CRD and set the `anonymous-auth` is set to `false`.

Default Value:

By default, anonymous access is enabled.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/docs/KubeletConfigDesign.md
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/#kubelet-authentication

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

4.2.2 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:

In OpenShift 4, the kublet config file is managed by the Machine Config Operator. By default, Unauthenticated/Unauthorized users have no access to OpenShift nodes. Run the following command:

```
# needs verification

#In one terminal, run:
  oc proxy

#Then in another terminal, run:
for name in $(oc get nodes -ojsonpath='{.items[*].metadata.name}')
do
        curl -sS http://127.0.0.1:8080/api/v1/nodes/$name/proxy/configz | jq -
r '.kubeletconfig.authorization.mode'
  done

# Alternative without oc proxy
POD=$(oc -n openshift-kube-apiserver get pod -l app=openshift-kube-apiserver
-o jsonpath='{.items[0].metadata.name}')

TOKEN=$(oc whoami -t)

for name in $(oc get nodes -ojsonpath='{.items[*].metadata.name}')
do
        oc exec -n openshift-kube-apiserver $POD -- curl -sS
https://172.25.0.1/api/v1/nodes/$name/proxy/configz -k -H "Authorization:
```

```
Bearer $TOKEN" | jq -r '.kubeletconfig.authorization.mode'
done
```

Verify that access is not successful.

Remediation:

None required. Unauthenticated/Unauthorized users have no access to OpenShift nodes.

Default Value:

By default, `--authorization-mode` argument is set to `Webhook`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/docs/KubeletConfigDesign.md
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/#kubelet-authentication

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

4.2.3 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:

OpenShift provides integrated management of certificates for internal cluster components. OpenShift 4 includes multiple CAs providing independent chains of trust, which ensure that a platform CA will never accidentally sign a certificate that can be used for the wrong purpose, increasing the security posture of the cluster. The Client CA location for the kubelet is defined in `/etc/kubernetes/kubelet.conf`.
Run the following command:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host grep -B3 clientCAFile:
/etc/kubernetes/kubelet.conf
done
```

Verify that the `clientCAFile` exists and is set to `/etc/kubernetes/kubelet-ca.crt`. The output should look like the following:

```
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  x509:
       clientCAFile: /etc/kubernetes/kubelet-ca.crt
```

Remediation:

None required. Changing the `clientCAFile` value is unsupported.

Default Value:

By default, `--client-ca-file` argument is set to `/etc/kubernetes/kubelet-ca.crt`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/docs/KubeletConfigDesign.md
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L28-L29
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/

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.

4.2.4 Verify that the read only port is not used or is set to 0 (Automated)

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:

In OpenShift 4, the kubelet is managed by the Machine Config Operator. The kubelet config file is found at `/etc/kubernetes/kubelet.conf`. OpenShift disables the read-only port (10255) on all nodes by setting the `kubelet-read-only-port` kubelet flag to 0 by default in OpenShift 4.6 and above. In OpenShift 4.5 and earlier, the `kubelet-read-only-port` argument is not used.
Run the following command:

```
# needs verification

oc -n openshift-kube-apiserver get cm kube-apiserver-pod -o yaml | grep --
color kubelet-read-only-port

oc -n openshift-kube-apiserver get cm config -o yaml | grep --color "kubelet-
read-only-port”
```

For OpenShift 4.5, verify that nothing is returned. Configuration information is available here: https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L135-L141
For OpenShift 4.6, verify that the `kubelet-read-only-port` is set to `0`.

Remediation:

In earlier versions of OpenShift 4, the `kubelet-read-only-port` argument is not used. Follow the instructions in the documentation to create a Kubelet config CRD and set the `--kubelet-read-only-port` is set to `0`.

Default Value:

By default, in OpenShift 4.5 and earlier, the `--kubelet-read-only-port` is not used. In OpenShift 4.6 and above, the `--kubelet-read-only-port` is set to `0`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/docs/KubeletConfigDesign.md
4. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.5/bindata/v4.1.0/config/defaultconfig.yaml#L28-L29
5. https://github.com/openshift/cluster-kube-apiserver-operator/blob/release-4.6/bindata/v4.1.0/config/defaultconfig.yaml#L114-L115
6. https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L135-L141
7. https://kubernetes.io/docs/reference/command-line-tools-reference/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.

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

Profile Applicability:

- Level 1

Description:

Do not disable timeouts on streaming connections.

Rationale:

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

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

Impact:

Long-lived connections could be interrupted.

Audit:

OpenShift uses the kubernetes default of 4 hours for the streaming-connection-idle-timeout argument. Unless the cluster administrator has added the value to the node configuration, the default will be used. The value is a timeout for HTTP streaming sessions going through a kubelet, like the port-forward, exec, or attach pod operations. The `streaming-connection-idle-timeout` should not be disabled by setting it to `zero`, but it can be lowered. Note that if the value is set too low, then users using those features may experience a service interruption due to the timeout.
The kubelet configuration is currently serialized as an ignition configuration, so it can be directly edited. However, there is also a new `kubelet-config-controller` added to the Machine Config Controller (MCC). This allows you to create a `KubeletConfig` custom resource (CR) to edit the kubelet parameters.
Run the following command on each node:

```
# needs verification

# Should return 1 for each node
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
```

```
        oc debug node/${node} -- chroot /host ps -ef | grep kubelet | grep
streaming-connection-idle-timeout
        echo $?
done

# Should return 1 for each node
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host grep
streamingConnectionIdleTimeout /etc/kubernetes/kubelet.conf
        echo $?
done
```

Verify that the `--streaming-connection-idle-timeout` argument is not set to `0`.
If the argument is not present, and there is a Kubelet config file specified by `--config`, check that it does not set `streamingConnectionIdleTimeout` to `0`.

Remediation:

Follow the instructions in the documentation to create a Kubelet config CRD and set the `--streaming-connection-idle-timeout` to the desired value. Do not set the value to `0`.

Default Value:

By default, `--streaming-connection-idle-timeout` is set to 4 hours.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
4. https://github.com/kubernetes/kubernetes/pull/18552

CIS Controls:

Version 6

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

4.2.6 Ensure that the --protect-kernel-defaults argument is not set (Automated)

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:

The OpenShift 4 kubelet modifies the system tunable; using the `protect-kernel-defaults` flag will cause the kubelet to fail on start if the tunables don't match the kubelet configuration and the OpenShift node will fail to start.
Run the following command:

```
# needs verification
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}'); do oc
debug node/${node} -- chroot /host more /etc/kubernetes/kubelet.conf; done
```

Verify that protectKernelDefaults is not present.

Remediation:

None required. The OpenShift 4 kubelet modifies the system tunable; using the `protect-kernel-defaults` flag will cause the kubelet to fail on start if the tunables don't match the kubelet configuration and the OpenShift node will fail to start.

Default Value:

By default, OpenShift 4 kubelet modifies the system tunable.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L618-L626
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

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

4.2.7 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:

OpenShift sets the `make-iptables-util-changes` argument to true by default.
Run the following command:

```
# needs verification

/bin/bash
flag=make-iptables-util-chains
opt=makeIPTablesUtilChains

# look at each machineconfigpool

while read -r pool nodeconfig; do
        # true by default
        value='true'
        # first look for the flag
        oc get machineconfig $nodeconfig -o json | jq -r
'.spec.config.systemd[][] | select(.name=="kubelet.service") | .contents' |
sed -n "/^ExecStart=/,/^\$/ { /^\\s*--$flag=false/ q 100 }"
        # if the above command exited with 100, the flag was false
        [ $? == 100 ] && value='false'
        # now look in the yaml KubeletConfig
```

```
        yamlconfig=$(oc get machineconfig $nodeconfig -o json | jq -r
'.spec.config.storage.files[] | select(.path=="/etc/kubernetes/kubelet.conf")
| .contents.source ' | sed 's/^data:,//' | while read; do echo -e
${REPLY//%/\\x}; done)
        echo "$yamlconfig" | sed -n "/^$opt:\\s*false\\s*$/ q 100"
        [ $? == 100 ] && value='false'
        echo "Pool $pool has $flag ($opt) set to $value"
done < <(oc get machineconfigpools -o json | jq -r '.items[] |
select(.status.machineCount>0) | .metadata.name + " " +
.spec.configuration.name')
```

Verify the `--make-iptables-util-chains` argument is set to true for each `machinepool`. For example:

```
Pool master has make-iptables-util-chains (makeIPTablesUtilChains) set to
true
Pool worker has make-iptables-util-chains (makeIPTablesUtilChains) set to
true
```

Remediation:

None required. The `--make-iptables-util-chains` argument is set to true by default.

Default Value:

By default, in OpenShift 4, the `--make-iptables-util-chains` argument is set to `true`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L618-L626
3. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

CIS Controls:

Version 6

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

Limitation and Control of Network Ports, Protocols, and Services

4.2.8 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:

In OpenShift 4, the `--hostname-override` argument is not used.
Run the following command on each machine pool. For example:

```
# needs verification

oc get machineconfig 01-worker-kubelet -o yaml | grep hostname-override
oc get machineconfig 01-master-kubelet -o yaml | grep hostname-override
```

Verify that `--hostname-override` argument does not exist.

Remediation:

None required.

Default Value:

By default, `--hostname-override` argument is not set.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/master/01-master-kubelet/_base/files/kubelet.yaml
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/worker/01-worker-kubelet/_base/files/kubelet.yaml
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
5. https://github.com/kubernetes/kubernetes/issues/22063

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

4.2.9 Ensure that the kubeAPIQPS [--event-qps] argument is set to 0 or a level which ensures appropriate event capture (Manual)

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:

OpenShift uses the `kubeAPIQPS` argument and sets it to a default value of `50`. When this value is set to > 0, event creations per second are limited to the value set. If this value is set to `0`, event creations per second are unlimited.
Run the following command on each machine pool. For example:

```
# needs verification

for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}'); do oc
debug node/${node} -- chroot /host more /etc/kubernetes/kubelet.conf; done

oc get machineconfig 01-worker-kubelet -o yaml | grep --color
kubeAPIQPS%3A%2050

oc get machineconfig 01-master-kubelet -o yaml | grep --color
kubeAPIQPS%3A%2050
```

Review the value set for the `kubeAPIQPS` argument and determine whether this has been set to an appropriate level for the cluster. If this value is set to `0`, event creations per second are unlimited.

Remediation:

Follow the documentation to edit kubelet parameters
https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters

```
KubeAPIQPS: <QPS>
```

Default Value:

By default, the `kubeAPIQPS` argument is set to `50`.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://docs.openshift.com/container-platform/4.5/scalability_and_performance/recommended-host-practices.html#create-a-kubeletconfig-crd-to-edit-kubelet-parameters_
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/master/01-master-kubelet/_base/files/kubelet.yaml
4. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/worker/01-worker-kubelet/_base/files/kubelet.yaml
5. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
6. https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/apis/kubeletconfig/v1beta1/types.go

CIS Controls:

Version 6

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

4.2.10 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:

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.

Impact:

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

Audit:

By default, OpenShift uses X.509 certificates to provide secure connections between the API server and `node/kubelet`. OpenShift Container Platform monitors certificates for proper validity, for the cluster certificates it issues and manages. The OpenShift Container Platform manages certificate rotation and the alerting framework has rules to help identify when a certificate issue is about to occur.
Run the following command on each node:

```
# needs verification
oc get configmap config -n openshift-kube-apiserver -ojson | jq -r
'.data["config.yaml"]' | jq '.kubeletClientInfo'
```

Verify that the `kubelet-client-certificate` argument is set to `/etc/kubernetes/static-pod-certs/secrets/kubelet-client/tls.crt`
Verify that the `kubelet-client-key` argument is set to `/etc/kubernetes/static-pod-certs/secrets/kublet-client/tls.key`

Remediation:

OpenShift automatically manages TLS authentication for the API server communication with the `node/kublet`. This is not configurable.

Default Value:

By default, OpenShift uses X.509 certificates to provide secure connections between the API server and `node/kubelet`. OpenShift does not use values assigned to the `tls-cert-file` or `tls-private-key-file` flags.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
3. https://rootsquash.com/2016/05/10/securing-the-kubernetes-api/
4. https://github.com/kelseyhightower/docker-kubernetes-tls-guide
5. https://jvns.ca/blog/2017/08/05/how-kubernetes-certificates-work/

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.

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

Profile Applicability:

- Level 1

Description:

Enable kubelet client certificate rotation.

Rationale:

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

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

Impact:

None

Audit:

This feature also requires the `RotateKubeletClientCertificate` feature gate to be enabled. The feature gate is enabled by default.
Run the following commands:

```
# needs verification

#Verify the rotateKubeletClientCertificate feature gate is not set to false
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot /host cat /etc/kubernetes/kubelet.conf
| grep RotateKubeletClientCertificate
done

# Verify the rotateCertificates argument is set to true
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot host grep rotate
/etc/kubernetes/kubelet.conf;
done
```

Verify that the `rotateKubeletClientCertificates` feature gate argument is not set to `false`.
Verify that the `rotateCertificates` argument is set to `true`.

Remediation:

None required.

Default Value:

By default, in OpenShift 4, kubelet client certificate rotation is enabled.

References:

1. https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html#understanding-machine-config-operator_control-plane
2. https://github.com/openshift/kubernetes-kubelet/blob/origin-4.5-kubernetes-1.18.3/config/v1beta1/types.go#L172-L181
3. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/master/01-master-kubelet/_base/files/kubelet.yaml
4. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/worker/01-worker-kubelet/_base/files/kubelet.yaml
5. https://github.com/kubernetes/kubernetes/pull/41912
6. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-tls-bootstrapping/#kubelet-configuration
7. https://kubernetes.io/docs/imported/release/notes/
8. https://kubernetes.io/docs/reference/command-line-tools-reference/feature-gates/

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.

4.2.12 Verify that the RotateKubeletServerCertificate argument is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Enable kubelet server certificate rotation.

Rationale:

`RotateKubeletServerCertificate` 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:

Run the following command on each node:

```
# needs verification

#Verify the rotateKubeletServerCertificate feature gate is on
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}'); do oc
debug node/${node} -- chroot /host more /etc/kubernetes/kubelet.conf; done

# Verify the rotateCertificates argument is set to true
for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}')
do
        oc debug node/${node} -- chroot host grep rotate
/etc/kubernetes/kubelet.conf;
done
```

Verify that the `RotateKubeletServerCertificate` argument is set to `true`.
Verify that the `rotateCertificates` argument is set to `true`

Remediation:

None required.

Default Value:

By default, kubelet server certificate rotation is disabled.

References:

1. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/master/01-master-kubelet/_base/files/kubelet.yaml
2. https://github.com/openshift/machine-config-operator/blob/release-4.5/templates/worker/01-worker-kubelet/_base/files/kubelet.yaml
3. https://github.com/kubernetes/kubernetes/pull/45059
4. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-tls-bootstrapping/#kubelet-configuration

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.

4.2.13 Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that the Kubelet is configured to only use strong cryptographic ciphers.

Rationale:

TLS ciphers have had a number of known vulnerabilities and weaknesses, which can reduce the protection provided by them. By default Kubernetes supports a number of TLS ciphersuites including some that have security concerns, weakening the protection provided.

Impact:

Kubelet clients that cannot support modern cryptographic ciphers will not be able to make connections to the Kubelet API.

Audit:

The set of cryptographic ciphers currently considered secure is the following:

```
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_GCM_SHA256
```

Ciphers for the API servers, authentication and the ingress controller can be configured using the `tlsSecurityProfile` parameter as of OpenShfit 4.3. The ingress controller provides external access to the API server. There are four TLS security profile types:

- Old
- Intermediate
- Modern
- Custom

Only the Old, Intermediate and Custom profiles are supported at this time for the Ingress controller. Custom provides the ability to specify individual TLS security profile parameters. Follow the steps in the documentation to configure the cipher suite for Ingress, API server and Authentication. https://docs.openshift.com/container-platform/4.5/networking/ingress-operator.html#nw-ingress-controller-configuration-parameters_configuring-ingress
Run the following commands to verify the cipher suite and minTLSversion for the ingress operator, authentication operator, `cliconfig`, OpenShift `APIserver` and Kube APIserver.

```
# needs verification

# verify cipher suites
oc describe --namespace=openshift-ingress-operator ingresscontroller/default

oc get kubeapiservers.operator.openshift.io cluster -o json |jq
.spec.observedConfig.servingInfo

oc get openshiftapiservers.operator.openshift.io cluster -o json |jq
.spec.observedConfig.servingInfo

oc get cm -n openshift-authentication v4-0-config-system-cliconfig -o
jsonpath='{.data.v4\-0\-config\-system\-cliconfig}' | jq .servingInfo

#check value for tlsSecurityProfile; null is returned if default is used
oc get kubeapiservers.operator.openshift.io cluster -o json |jq
.spec.tlsSecurityProfile
```

Verify that the cipher suites are appropriate.
Verify that the `tlsSecurityProfile` is set to the value you chose.
Note: The HAProxy Ingress controller image does not support TLS 1.3 and because the Modern profile requires TLS 1.3, it is not supported. The Ingress Operator converts the Modern profile to Intermediate. The Ingress Operator also converts the TLS 1.0 of an Old or Custom profile to 1.1, and TLS 1.3 of a Custom profile to 1.2.

Remediation:

Follow the directions above and in the OpenShift documentation to configure the `tlsSecurityProfile`. Configuring Ingress

Default Value:

By default the Kubernetes API server supports a wide range of TLS ciphers

CIS Controls:

Version 6

3.4 Use Only Secure Channels For Remote System Administration

Perform all remote administration of servers, workstation, network devices, and similar equipment over secure channels. Protocols such as telnet, VNC, RDP, or others that do not actively support strong encryption should only be used if they are performed over a secondary encryption channel, such as SSL, TLS or IPSEC.

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

Use multi-factor authentication and encrypted channels for all administrative account access.

5 Policies

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

5.1 RBAC and Service Accounts

5.1.1 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:

OpenShift provides a set of default cluster roles that you can bind to users and groups cluster-wide or locally (per project namespace). Be mindful of the difference between local and cluster bindings. For example, if you bind the cluster-admin role to a user by using a local role binding, it might appear that this user has the privileges of a cluster administrator. This is not the case. Binding the cluster-admin to a user in a project grants super administrator privileges for only that project to the user. You can use the oc CLI to view cluster roles and bindings by using the oc describe command. For more information, see Default Cluster Roles
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.
Review users and groups bound to cluster-admin and decide whether they require such access. Consider creating least-privilege roles for users and service accounts.
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.

```
# needs verification

# To get a list of users and service accounts with the cluster-admin role
oc get clusterrolebindings -o=custom-
columns=NAME:.metadata.name,ROLE:.roleRef.name,SUBJECT:.subjects[*].kind |
grep cluster-admin

# To verity that kbueadmin is removed, no results should be returned
oc get secrets kubeadmin -n kube-system
```

Review each principal listed and ensure that cluster-admin privilege is required for it.
Verify that the kubeadmin user no longer exists.

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 :

```
oc 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/reference/access-authn-authz/rbac/#user-facing-roles

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.

5.1.2 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 in a OpenShift cluster the following list of principals have get privileges on secret objects

```
for i in $(oc get clusterroles -o jsonpath='{.items[*].metadata.name}'); do
oc describe clusterrole ${i}; done



# The following default cluster roles have get privileges on secret objects
```

```
admin

cloud-credential-operator-role

cluster-admin

cluster-image-registry-operator

cluster-monitoring-operator

cluster-node-tuning-operator

edit

kube-state-metrics

machine-config-controller

marketplace-operator

openshift-ingress-operator

prometheus-operator

registry-admin

registry-editor

system:aggregate-to-edit

system:controller:expand-controller

system:controller:generic-garbage-collector

system:controller:namespace-controller

system:controller:operator-lifecycle-manager

system:controller:persistent-volume-binder

system:kube-controller-manager

system:master

system:node

system:openshift:controller:build-controller

system:openshift:controller:cluster-quota-reconciliation-controller

system:openshift:controller:ingress-to-route-controller

system:openshift:controller:service-ca

system:openshift:controller:service-serving-cert-controller
```

```
system:openshift:controller:serviceaccount-pull-secrets-controller

system:openshift:controller:template-service-broker
```

5.1.3 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

```
# needs verification

oc get roles --all-namespaces -o yaml

for i in $(oc get roles -A -o jsonpath='{.items[*].metadata.name}'); do oc
describe clusterrole ${i}; done
```

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

```
oc get clusterroles -o yaml

for i in $(oc get clusterroles -o jsonpath='{.items[*].metadata.name}'); do
oc describe clusterrole ${i}; done
```

Remediation:

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

5.1.4 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 in a kubeadm cluster the following list of principals have `create` privileges on `pod` objects

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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

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

Impact:

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

Audit:

Every OpenShift project has its own service accounts. Every service account has an associated user name that can be granted roles, just like a regular user. The user name for each service account is derived from its project and the name of the service account. Service accounts are required in each project to run builds, deployments, and other pods. The default service accounts that are automatically created for each project are isolated by the project namespace.

Remediation:

None required.

Default Value:

By default, in OpenShift 4 every project has its own service accounts. Every service account has an associated user name that can be granted roles, just like a regular user. The user name for each service account is derived from its project and the name of the service account. Service accounts are required in each project to run builds, deployments, and other pods. The default service accounts that are automatically created for each project are isolated by the project namespace.

References:

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

5.1.6 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.

5.2 Pod Security Policies

A Pod Security Policy (SCC) is a cluster-level resource that controls security settings for pods. PodSecurityPolicies are used in conjunction with the PodSecurityPolicy admission controller plugin.

OpenShift uses the Security Context Constraint (SCC) admission controller plugin instead of PodSecurityPolicies. The SCC plugin cannot be disabled. The PSP plugin cannot be enabled.

Similar to the way that RBAC resources control user access, administrators can use Security Context Constraints (SCCs) to control permissions for pods. These permissions include actions that a pod, a collection of containers, can perform and what resources it can access. You can use SCCs to define a set of conditions that a pod must run with in order to be accepted into the system.

By default, OpenShift 4 is configured with multiple SCCs. You can query SCCs with the following command:

```
oc get scc
```

5.2.1 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 Security Context Constraint (SCC) defined which does not permit privileged containers.

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

Impact:

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

Audit:

The set of SCCs that admission uses to authorize a pod are determined by the user identity and groups that the user belongs to. Additionally, if the pod specifies a service account, the set of allowable SCCs includes any constraints accessible to the service account.
Admission uses the following approach to create the final security context for the pod:

- Retrieve all SCCs available for use.
- Generate field values for security context settings that were not specified on the request.
- Validate the final settings against the available constraints.

If a matching set of constraints is found, then the pod is accepted. If the request cannot be matched to an SCC, the pod is rejected.
A pod must validate every field against the SCC.
Get the set of SCCs with the following command:

```
oc get scc
```

For each SCC, check whether privileged is enabled:

```
# needs verification
for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Allow Privileged"; done
```

Verify that there is at least one SCC which does not have `Allow Privileged` set to `true`.

Remediation:

Create a SCC as described in the OpenShift documentation, ensuring that the `Allow Privileged` field is set to `false`.

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid.            Allow Privileged: false

hostaccess.        Allow Privileged: false
```

```
Hostmount-anyuid   Allow Privileged: false

hostnetwork        Allow Privileged: false

node-exporter      Allow Privileged: true

non-root           Allow Privileged: false

Privileged         Allow Privileged: true

Restricted         Allow Privileged: false
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. 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.

5.2.2 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 Security Context Constraint (SCC) 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 SCC and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that SCC.

Impact:

Pods defined with `Allow Host PID: true` will not be permitted unless they are run under a specific SCC.

Audit:

Get the set of SCCs with the following command:

```
oc get scc
```

For each SCC, check whether Allow Host PID is true:

```
for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Allow Host PID"; done
```

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

Remediation:

Create a SCC as described in the OpenShift documentation, ensuring that the `Allow Host PID` field is set to false.

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid                Allow Host PID: false

hostaccess            Allow Host PID: true

Hostmount-anyuid      Allow Host PID: false

hostnetwork           Allow Host PID: false

node-exporter         Allow Host PID: true

non-root              Allow Host PID: false

privileged            Allow Host PID: false

restricted            Allow Host PID: false
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://kubernetes.io/docs/concepts/policy/pod-security-policy/

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.

5.2.3 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 Security Context Constraint (SCC) 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 SCC and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that SCC.

Impact:

Pods defined with `Allow Host IPC: true` will not be permitted unless they are run under a specific SCC.

Audit:

Get the set of SCCs with the following command:

```
oc get scc
```

For each SCC, check whether `Allow Host IPC` is enabled:

```
for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Allow Host IPC"; done
```

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

Remediation:

Create a SCC as described in the OpenShift documentation, ensuring that the `Allow Host IPC` field is set to `false`.

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid                  Allow Host IPC: false

hostaccess              Allow Host IPC: true

Hostmount-anyuid        Allow Host IPC: false

hostnetwork             Allow Host IPC: false

node-exporter           Allow Host IPC: false

non-root                Allow Host IPC: false

privileged              Allow Host IPC: false

restricted              Allow Host IPC: false
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://kubernetes.io/docs/concepts/policy/pod-security-policy/

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.

5.2.4 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 Security Context Constraint (SCC) 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 SCC and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that SCC.

Impact:

Pods defined with `Allow Host Network: true` will not be permitted unless they are run under a specific SCC.

Audit:

Get the set of SCCs with the following command:

```
oc get scc
```

For each SCC, check whether `Allow Host Network` is enabled:

```
for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Allow Host Network"; done
```

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

Remediation:

Create a SCC as described in the OpenShift documentation, ensuring that the `Allow Host Network` field is omitted or set to `false`.

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid                  Allow Host Network: false

hostaccess              Allow Host Network: true

hostmount-anyuid.       Allow Host Network: false

hostnetwork             Allow Host Network: true

node-exporter           Allow Host Network: true

non-root                Allow Host Network: false

privileged              Allow Host Network: true

restricted              Allow Host Network: false
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://kubernetes.io/docs/concepts/policy/pod-security-policy/

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.

5.2.5 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 Security Context Constraint (SCC) 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 SCC and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that SCC.

Impact:

Pods defined with `Allow Privilege Escalation: true` will not be permitted unless they are run under a specific SCC.

Audit:

Get the set of SCCs with the following command:

```
oc get scc
```

For each SCC, check whether privileged is enabled:

```
# needs verification
for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Allow Privilege Escalation"; done
```

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

Remediation:

Create a SCC as described in the OpenShift documentation, ensuring that the `Allow Privilege Escalation` field is set to `false`.

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid                Allow Privilege Escalation: true

hostaccess            Allow Privilege Escalation: true

hostmount-anyuid      Allow Privilege Escalation: true

hostnetwork           Allow Privilege Escalation: true

node-exporter         Allow Privilege Escalation: true

non-root              Allow Privilege Escalation: true

privileged            Allow Privilege Escalation: true

restricted            Allow Privilege Escalation: true
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://kubernetes.io/docs/concepts/policy/pod-security-policy/

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.

5.2.6 Minimize the admission of root containers (Manual)

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 an escalated likelihood of container breakout.

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

There should be at least one Security Context Constraint (SCC) defined which does not permit root users in a container.

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

Impact:

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

Audit:

Get the set of SCCs with the following command:

```
oc get scc
```

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

```
# needs verification

for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Run As User Strategy"; done

#For SCCs with MustRunAs verify that the range of UIDs does not include 0

for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "\sUID"; done
```

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

Remediation:

None required. By default, OpenShift includes the non-root SCC with the the `Run As User Strategy` is set to either `MustRunAsNonRoot`. If additional SCCs are appropriate, follow the OpenShift documentation to create custom SCCs.

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid                  Run As User Strategy: RunAsAny

hostaccess              Run As User Strategy: MustRunAsRange

hostmount-anyuid        Run As User Strategy: RunAsAny

hostnetwork             Run As User Strategy: MustRunAsRange

node-exporter           Run As User Strategy: RunAsAny

non-root                Run As User Strategy: MustRunAsNonRoot

privileged              Run As User Strategy: RunAsAny

restricted              Run As User Strategy: MustRunAsRange
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://kubernetes.io/docs/concepts/policy/pod-security-policy/

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.

5.2.7 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 Security Context Constraint (SCC) 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 SCC and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that SCC.

Impact:

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

Audit:

Get the set of SCCs with the following command:

```
oc get scc
```

For each SCC, check whether `NET_RAW` is disabled:

```
# needs verification
for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Required Drop Capabilities"; done
```

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

Remediation:

Create a SCC as described in the OpenShift documentation, ensuring that the `Required Drop Capabilities` is set to include either `NET_RAW` or `ALL`.

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid                  Required Drop Capabilities: MKNOD

hostaccess              Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID

hostmount-anyuid        Required Drop Capabilities: MKNOD

hostnetwork             Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID

node-exporter           Required Drop Capabilities: <none>

non-root                Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID

privileged              Required Drop Capabilities: <none>

restricted              Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
3. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

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.

5.2.8 Minimize the admission of containers with added capabilities (Manual)

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 Security Context Constraint (SCC) 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 SCC and you should carefully check RBAC controls to ensure that only limited service accounts and users are given permission to access that SCC.

Impact:

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

Audit:

Get the set of SCCs with the following command:

```
oc get scc
```

For each SCC, check the values for `Allowed Capabilities`:

```
# needs verification
oc describe scc <name> | grep “Default Add Capabilities”

for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Allowed Capabilities"; done

for i in `oc get scc --template '{{range
```

```
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Default Add Capabilities"; done
```

Minimize the number of SCCs that have `Allowed Capabilities` set to anything other than an empty array.
Minimize the number of SCCs that have `Default Add Capabilities` set to anything other than an empty array.

Remediation:

Ensure that `Allowed Capabilities` is set to an empty array for every SCC in the cluster except for the `privileged` SCC.

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid                  Allowed Capabilities: <none>

hostaccess              Allowed Capabilities: <none>

hostmount-anyuid        Allowed Capabilities: <none>

hostnetwork             Allowed Capabilities: <none>

node-exporter           Allowed Capabilities: <none>

non-root                Allowed Capabilities: <none>

privileged              Allowed Capabilities: *

restricted              Allowed Capabilities: <none>


anyuid                  Default Add Capabilities: <none>

hostaccess              Default Add Capabilities: <none>

hostmount-anyuid        Default Add Capabilities: <none>

hostnetwork             Default Add Capabilities: <none>

node-exporter           Default Add Capabilities: <none>

non-root                Default Add Capabilities: <none>

privileged              Default Add Capabilities: <none>

restricted              Default Add Capabilities: <none>
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
3. https://www.nccgroup.com/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

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.

5.2.9 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 which require capabilities to operate will not be permitted.

Audit:

Get the set of SCCs with the following command:

```
oc get scc
```

For each SCC, check whether capabilities have been forbidden:

```
# needs verification
oc describe scc <name> | grep “Required Drop Capabilities”

for i in `oc get scc --template '{{range
.items}}{{.metadata.name}}{{"\n"}}{{end}}'`; do echo "$i"; oc describe scc $i
| grep "Required Drop Capabilities"; done
```

Remediation:

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

Default Value:

By default, OpenShift 4 clusters include the following SCCs:

```
anyuid                  Required Drop Capabilities: MKNOD

hostaccess              Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID

hostmount-anyuid        Required Drop Capabilities: MKNOD

hostnetwork             Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID

node-exporter           Required Drop Capabilities: <none>

non-root                Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID

privileged              Required Drop Capabilities: <none>

restricted              Required Drop Capabilities: KILL,MKNOD,SETUID,SETGID
```

References:

1. https://docs.openshift.com/container-platform/4.5/authentication/managing-security-context-constraints.html
2. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
3. https://www.nccgroup.com/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

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.

5.3 Network Policies and CNI

5.3.1 Ensure that the CNI in use supports Network Policies (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 Ingress and Egress network policies.
OpenShift Container Platform uses a software-defined networking (SDN) approach to provide a unified cluster network that enables communication between Pods across the OpenShift Container Platform cluster. This Pod network is established and maintained by the OpenShift SDN, which configures an overlay network using Open vSwitch (OVS). The OpenShift SDN uses Network Policies. The OpenShift SDN CNI plug-in provides all Kubernetes v1 NetworkPolicy features except for egress policy types and IPBlock. However, OpenShift provides means to implement fine grained filtering of egress traffic. OpenShift provides several options for controlling the traffic leaving the cluster. These options are :

- Egress firewall
- Egress routers
- Egress static IP

Remediation:

None required.

Default Value:

This will depend on the CNI plugin in use.

References:

1. https://docs.openshift.com/container-platform/4.5/networking/openshift-sdn/about-openshift-sdn.html
2. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/

5.3.2 Ensure that all Namespaces have Network Policies defined (Automated)

Profile Applicability:

- Level 2

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:

The OpenShift 4 CNI plugin uses network policies and by default all Pods in a project are accessible from other Pods and network endpoints. To isolate one or more Pods in a project, you create NetworkPolicy objects in that project to indicate the allowed incoming connections. Project administrators can create and delete NetworkPolicy objects within their own project. For more information see:
Run the following command and review the `NetworkPolicy` objects created in the cluster.

```
oc -n all get networkpolicy
```

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.

Default Value:

By default, all Pods in a project are accessible from other Pods and network endpoints; network policies are not created.

References:

1. https://docs.openshift.com/container-platform/4.5/networking/network_policy/about-network-policy.html
2. https://docs.openshift.com/container-platform/4.5/networking/network_policy/creating-network-policy.html
3. https://docs.openshift.com/container-platform/4.5/networking/network_policy/multitenant-network-policy.html
4. https://docs.openshift.com/container-platform/4.5/networking/network_policy/default-network-policy.html
5. https://kubernetes.io/docs/concepts/services-networking/network-policies/
6. https://octetz.com/docs/2019/2019-04-22-netpol-api-k8s/
7. https://kubernetes.io/docs/tasks/administer-cluster/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.

5.4 Secrets Management

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Impact:

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

Audit:

Information about ways to provide sensitive data to pods is included in the documentation.
Providing sensitive data to pods
Run the following command to find references to objects which use environment variables defined from secrets.

```
oc 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, application secrets are not defined.

In a default OpenShift 4 cluster, the following platform objects are returned

```
Pod image-registry-f56c674f-qp8f2

Pod image-registry-f56c674f-w8fck

Pod router-default-7856544cc7-bhspv

Pod router-default-7856544cc7-tq6k7

Deployment image-registry

Deployment router-default

ReplicaSet image-registry-6dd744f76b

ReplicaSet image-registry-f56c674f

ReplicaSet router-default-7856544cc7
```

References:

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

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.

5.4.2 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 secrets are used across both Kubernetes and non-Kubernetes environments.

Impact:

None

Audit:

OpenShift supports a broad ecosystem of security partners many of whom provide integration with enterprise secret vaults.
Review your secrets management implementation.

Remediation:

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

Default Value:

By default, no external secret management is configured.

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.

5.5 Extensible Admission Control

5.5.1 Configure Image Provenance using image controller configuration parameters (Manual)

Profile Applicability:

- Level 2

Description:

Configure Image Provenance for your deployment.

Rationale:

Kubernetes supports plugging in provenance rules to accept or reject the images in your deployments. You could configure such rules to ensure that only approved images are deployed in the cluster.

You can control which images can be imported, tagged, and run in a cluster using the image controller. For additional information on the image controller, see Image configuration resources

Impact:

You need to regularly maintain your provenance configuration based on container image updates.

Audit:

Review the image controller parameters in your cluster and verify that image provenance is configured as appropriate.

Remediation:

Follow the OpenShift documentation: [Image configuration resources](https://docs.openshift.com/container-platform/4.5/openshift_images/image-configuration.html

Default Value:

By default, image provenance is not set.

References:

1. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#imagepolicywebhook
2. https://github.com/kubernetes/community/blob/master/contributors/design-proposals/image-provenance.md
3. https://hub.docker.com/r/dnurmi/anchore-toolbox/
4. https://github.com/kubernetes/kubernetes/issues/22888

CIS Controls:

Version 6

18 Application Software Security
Application Software Security

5.7 General Policies

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

5.7.1 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:

OpenShift Projects wrap Kubernetes namespaces and are used by default in OpenShift 4. Run the following command and review the namespaces created in the cluster.

```
oc 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. `openshift` -
4. `openshift-*` - The namespace for objects created by OpenShift

References:

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

CIS Controls:

Version 6

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

5.7.2 Ensure that the seccomp profile is set to docker/default in your pod definitions (Manual)

Profile Applicability:

- Level 2

Description:

Enable `default` seccomp profile in your pod definitions.

Rationale:

Seccomp (secure computing mode) is used to restrict the set of system calls applications can make, allowing cluster administrators greater control over the security of workloads running in the cluster. Kubernetes disables seccomp profiles by default for historical reasons. You should enable it to ensure that the workloads have restricted actions available within the container.

Impact:

If the `default` seccomp profile is too restrictive for you, you will need to create and manage your own seccomp profiles.

Audit:

In OpenShift 4, CRI-O is the supported runtime. CRI-O runs unconfined by default in order to meet CRI conformance criteria.
On RHEL CoreOS, the default seccomp policy is associated with CRI-O and stored in `/etc/crio/seccomp.json`. The default profile is applied when the user asks for the runtime/default profile via annotation to the pod and when the associated SCC allows use of the specified seccomp profile.
Configuration of allowable seccomp profiles is managed through OpenShift Security Context Constraints.

Remediation:

To enable the `default` seccomp profile, use the reserved value `/runtime/default` that will make sure that the pod uses the default policy available on the host.

Default Value:

By default, seccomp profile is set to `unconfined` which means that no seccomp profiles are enabled.

References:

1. http://finder.cox.net/main?ParticipantID=96e687opkbv4scrood8k84drs6gw5duf&FailedURI=http%3A%2F%2Fpkgs.devel.redhat.com%2Fcgit%2Frpms%2Fcri-o%2Ftree%2Fseccomp.json%3Fh%3Drhaos-4.6-rhel-8&FailureMode=1&Implementation=&AddInType=4&Version=pywr1.0&ClientLocation=us
2. https://docs.openshift.com/container-platform/4.2/authentication/managing-security-context-constraints.html#security-context-constraints-about_configuring-internal-oauth
3. https://github.com/kubernetes/kubernetes/issues/39845
4. https://github.com/kubernetes/kubernetes/pull/21790
5. https://github.com/kubernetes/community/blob/master/contributors/design-proposals/seccomp.md#examples
6. https://docs.docker.com/engine/security/seccomp/

CIS Controls:

Version 6

5 Controlled Use of Administration Privileges
Controlled Use of Administration Privileges

5.7.3 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.
OpenShift's Security Context Constraint feature is on by default in OpenShift 4 and applied to all pods deployed. SCC selection is determined by a combination of the values in the securityContext and the rolebindings for the account deploying the pod.

Remediation:

Follow the Kubernetes documentation and apply security contexts to your pods. For a suggested list of security contexts, you may refer to the CIS Security Benchmark for Docker Containers.

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

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

5.7.4 The default namespace should not be used (Automated)

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:

In OpenShift, projects (namespaces) are used to group and isolate related objects. When a request is made to create a new project using the web console or oc new-project command, an endpoint in OpenShift Container Platform is used to provision the project according to a template, which can be customized.
The cluster administrator can allow and configure how developers and service accounts can create, or self-provision, their own projects. Regular users do not have access to the default project.
Projects starting with openshift- and kube- host cluster components that run as Pods and other infrastructure components. As such, OpenShift does not allow you to create Projects starting with openshift- or kube- using the oc new-project command.
For more information, see
Working with projects and
Configuring project creation
Run this command to list objects in default namespace

```
oc project default
oc get all
```

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

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Control Plane Components**		
1.1	**Master Node Configuration Files**		
1.1.1	Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.2	Ensure that the API server pod specification file ownership is set to root:root (Automated)	☐	☐
1.1.3	Ensure that the controller manager pod specification file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.4	Ensure that the controller manager pod specification file ownership is set to root:root (Automated)	☐	☐
1.1.5	Ensure that the scheduler pod specification file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.6	Ensure that the scheduler pod specification file ownership is set to root:root (Automated)	☐	☐
1.1.7	Ensure that the etcd pod specification file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.8	Ensure that the etcd pod specification file ownership is set to root:root (Automated)	☐	☐
1.1.9	Ensure that the Container Network Interface file permissions are set to 644 or more restrictive (Manual)	☐	☐
1.1.10	Ensure that the Container Network Interface file ownership is set to root:root (Manual)	☐	☐
1.1.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive (Automated)	☐	☐
1.1.12	Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)	☐	☐
1.1.13	Ensure that the admin.conf file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.14	Ensure that the admin.conf file ownership is set to root:root (Automated)	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.16	Ensure that the scheduler.conf file ownership is set to root:root (Automated)	☐	☐
1.1.17	Ensure that the controller-manager.conf file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.18	Ensure that the controller-manager.conf file ownership is set to root:root (Automated)	☐	☐

1.1.19	Ensure that the OpenShift PKI directory and file ownership is set to root:root (Automated)	☐	☐
1.1.20	Ensure that the OpenShift PKI certificate file permissions are set to 644 or more restrictive (Manual)	☐	☐
1.1.21	Ensure that the OpenShift PKI key file permissions are set to 600 (Manual)	☐	☐
1.2	**API Server**		
1.2.1	Ensure that anonymous requests are authorized (Manual)	☐	☐
1.2.2	Ensure that the --basic-auth-file argument is not set (Automated)	☐	☐
1.2.3	Ensure that the --token-auth-file parameter is not set (Automated)	☐	☐
1.2.4	Use https for kubelet connections (Automated)	☐	☐
1.2.5	Ensure that the kubelet uses certificates to authenticate (Automated)	☐	☐
1.2.6	Verify that the kubelet certificate authority is set as appropriate (Automated)	☐	☐
1.2.7	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	☐	☐
1.2.8	Verify that the Node authorizer is enabled (Automated)	☐	☐
1.2.9	Verify that RBAC is enabled (Automated)	☐	☐
1.2.10	Ensure that the APIPriorityAndFairness feature gate is enabled (Manual)	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysAdmit is not set (Automated)	☐	☐
1.2.12	Ensure that the admission control plugin AlwaysPullImages is not set (Manual)	☐	☐
1.2.13	Ensure that the admission control plugin SecurityContextDeny is not set (Manual)	☐	☐
1.2.14	Ensure that the admission control plugin ServiceAccount is set (Automated)	☐	☐
1.2.15	Ensure that the admission control plugin NamespaceLifecycle is set (Automated)	☐	☐
1.2.16	Ensure that the admission control plugin SecurityContextConstraint is set (Automated)	☐	☐
1.2.17	Ensure that the admission control plugin NodeRestriction is set (Automated)	☐	☐
1.2.18	Ensure that the --insecure-bind-address argument is not set (Automated)	☐	☐
1.2.19	Ensure that the --insecure-port argument is set to 0 (Automated)	☐	☐
1.2.20	Ensure that the --secure-port argument is not set to 0 (Automated)	☐	☐

1.2.21	Ensure that the healthz endpoint is protected by RBAC (Automated)	☐	☐
1.2.22	Ensure that the --audit-log-path argument is set (Automated)	☐	☐
1.2.23	Ensure that the audit logs are forwarded off the cluster for retention (Automated)	☐	☐
1.2.24	Ensure that the maximumRetainedFiles argument is set to 10 or as appropriate (Automated)	☐	☐
1.2.25	Ensure that the maximumFileSizeMegabytes argument is set to 100 or as appropriate (Automated)	☐	☐
1.2.26	Ensure that the --request-timeout argument is set as appropriate (Automated)	☐	☐
1.2.27	Ensure that the --service-account-lookup argument is set to true (Automated)	☐	☐
1.2.28	Ensure that the --service-account-key-file argument is set as appropriate (Automated)	☐	☐
1.2.29	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Automated)	☐	☐
1.2.30	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	☐	☐
1.2.31	Ensure that the --client-ca-file argument is set as appropriate (Automated)	☐	☐
1.2.32	Ensure that the --etcd-cafile argument is set as appropriate (Automated)	☐	☐
1.2.33	Ensure that the --encryption-provider-config argument is set as appropriate (Manual)	☐	☐
1.2.34	Ensure that encryption providers are appropriately configured (Manual)	☐	☐
1.2.35	Ensure that the API Server only makes use of Strong Cryptographic Ciphers (Manual)	☐	☐
1.3	**Controller Manager**		
1.3.1	Ensure that garbage collection is configured as appropriate (Manual)	☐	☐
1.3.2	Ensure that controller manager healthz endpoints are protected by RBAC (Automated)	☐	☐
1.3.3	Ensure that the --use-service-account-credentials argument is set to true (Automated)	☐	☐
1.3.4	Ensure that the --service-account-private-key-file argument is set as appropriate (Automated)	☐	☐
1.3.5	Ensure that the --root-ca-file argument is set as appropriate (Automated)	☐	☐
1.3.6	Ensure that the RotateKubeletServerCertificate argument is set to true (Automated)	☐	☐

1.3.7	Ensure that the --bind-address argument is set to 127.0.0.1 (Automated)	☐	☐
1.4	**Scheduler**		
1.4.1	Ensure that the healthz endpoints for the scheduler are protected by RBAC (Automated)	☐	☐
1.4.2	Verify that the scheduler API service is protected by authentication and authorization (Automated)	☐	☐
2	**etcd**		
2.1	Ensure that the --cert-file and --key-file arguments are set as appropriate (Automated)	☐	☐
2.2	Ensure that the --client-cert-auth argument is set to true (Automated)	☐	☐
2.3	Ensure that the --auto-tls argument is not set to true (Automated)	☐	☐
2.4	Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate (Automated)	☐	☐
2.5	Ensure that the --peer-client-cert-auth argument is set to true (Automated)	☐	☐
2.6	Ensure that the --peer-auto-tls argument is not set to true (Automated)	☐	☐
2.7	Ensure that a unique Certificate Authority is used for etcd (Manual)	☐	☐
3	**Control Plane Configuration**		
3.1	**Authentication and Authorization**		
3.1.1	Client certificate authentication should not be used for users (Manual)	☐	☐
3.2	**Logging**		
3.2.1	Ensure that a minimal audit policy is created (Automated)	☐	☐
3.2.2	Ensure that the audit policy covers key security concerns (Manual)	☐	☐
4	**Worker Nodes**		
4.1	**Worker Node Configuration Files**		
4.1.1	Ensure that the kubelet service file permissions are set to 644 or more restrictive (Automated)	☐	☐
4.1.2	Ensure that the kubelet service file ownership is set to root:root (Automated)	☐	☐
4.1.3	If proxy kubeconfig file exists ensure permissions are set to 644 or more restrictive (Automated)	☐	☐
4.1.4	If proxy kubeconfig file exists ensure ownership is set to root:root (Automated)	☐	☐
4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 644 or more restrictive (Automated)	☐	☐
4.1.6	Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root (Automated)	☐	☐

4.1.7	Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Automated)	☐	☐
4.1.8	Ensure that the client certificate authorities file ownership is set to root:root (Automated)	☐	☐
4.1.9	Ensure that the kubelet --config configuration file has permissions set to 644 or more restrictive (Automated)	☐	☐
4.1.10	Ensure that the kubelet configuration file ownership is set to root:root (Automated)	☐	☐
4.2	**Kubelet**		
4.2.1	Ensure that the --anonymous-auth argument is set to false (Automated)	☐	☐
4.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	☐	☐
4.2.3	Ensure that the --client-ca-file argument is set as appropriate (Automated)	☐	☐
4.2.4	Verify that the read only port is not used or is set to 0 (Automated)	☐	☐
4.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated)	☐	☐
4.2.6	Ensure that the --protect-kernel-defaults argument is not set (Automated)	☐	☐
4.2.7	Ensure that the --make-iptables-util-chains argument is set to true (Automated)	☐	☐
4.2.8	Ensure that the --hostname-override argument is not set (Manual)	☐	☐
4.2.9	Ensure that the kubeAPIQPS [--event-qps] argument is set to 0 or a level which ensures appropriate event capture (Manual)	☐	☐
4.2.10	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	☐	☐
4.2.11	Ensure that the --rotate-certificates argument is not set to false (Automated)	☐	☐
4.2.12	Verify that the RotateKubeletServerCertificate argument is set to true (Automated)	☐	☐
4.2.13	Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)	☐	☐
5	**Policies**		
5.1	**RBAC and Service Accounts**		
5.1.1	Ensure that the cluster-admin role is only used where required (Manual)	☐	☐
5.1.2	Minimize access to secrets (Manual)	☐	☐
5.1.3	Minimize wildcard use in Roles and ClusterRoles (Manual)	☐	☐
5.1.4	Minimize access to create pods (Manual)	☐	☐

5.1.5	Ensure that default service accounts are not actively used. (Automated)	☐	☐
5.1.6	Ensure that Service Account Tokens are only mounted where necessary (Manual)	☐	☐
5.2	**Pod Security Policies**		
5.2.1	Minimize the admission of privileged containers (Manual)	☐	☐
5.2.2	Minimize the admission of containers wishing to share the host process ID namespace (Automated)	☐	☐
5.2.3	Minimize the admission of containers wishing to share the host IPC namespace (Automated)	☐	☐
5.2.4	Minimize the admission of containers wishing to share the host network namespace (Automated)	☐	☐
5.2.5	Minimize the admission of containers with allowPrivilegeEscalation (Automated)	☐	☐
5.2.6	Minimize the admission of root containers (Manual)	☐	☐
5.2.7	Minimize the admission of containers with the NET_RAW capability (Manual)	☐	☐
5.2.8	Minimize the admission of containers with added capabilities (Manual)	☐	☐
5.2.9	Minimize the admission of containers with capabilities assigned (Manual)	☐	☐
5.3	**Network Policies and CNI**		
5.3.1	Ensure that the CNI in use supports Network Policies (Manual)	☐	☐
5.3.2	Ensure that all Namespaces have Network Policies defined (Automated)	☐	☐
5.4	**Secrets Management**		
5.4.1	Prefer using secrets as files over secrets as environment variables (Manual)	☐	☐
5.4.2	Consider external secret storage (Manual)	☐	☐
5.5	**Extensible Admission Control**		
5.5.1	Configure Image Provenance using image controller configuration parameters (Manual)	☐	☐
5.7	**General Policies**		
5.7.1	Create administrative boundaries between resources using namespaces (Manual)	☐	☐
5.7.2	Ensure that the seccomp profile is set to docker/default in your pod definitions (Manual)	☐	☐
5.7.3	Apply Security Context to Your Pods and Containers (Manual)	☐	☐
5.7.4	The default namespace should not be used (Automated)	☐	☐

Appendix: Change History

Date	Version	Changes for this version