CIS Center for Internet Security®

CIS Benchmarks™

CIS Kubernetes Benchmark

v1.9.0 - 03-25-2024

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

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

- Monitoring the base operating system for vulnerabilities and quickly updating with the latest security patches
- Monitoring applications and libraries for vulnerabilities and quickly updating with the latest security patches

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

This document provides prescriptive guidance for establishing a secure configuration posture for Kubernetes v1.27 - v1.29. To obtain the latest version of this guide, please visit www.cisecurity.org . If you have questions, comments, or have identified ways to improve this guide, please write us at support@cisecurity.org.
**Special Note: **The set of configuration files mentioned anywhere throughout this benchmark document may vary according to the deployment tool and the platform. Any reference to a configuration file should be modified according to the actual configuration files used on the specific deployment.
For example, the configuration file for the Kubernetes API server installed by the `kubeadm` tool may be found in `/etc/kubernetes/manifests/kube-apiserver.yaml`, but the same file may be called `/etc/kubernetes/manifests/kube-apiserver.manifest` when installed by `kops` or `kubespray`.

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 Kubernetes v1.27 - v1.29.

Consensus Guidance

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

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

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
`Monospace font`	Used for inline code, commands, 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

Recommendation Definitions

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

Title

Concise description for the recommendation's intended configuration.

Assessment Status

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

Automated

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

Manual

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

Profile

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

Description

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

Rationale Statement

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

Impact Statement

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

Audit Procedure

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

Remediation Procedure

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

Default Value

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

References

Additional documentation relative to the recommendation.

CIS Critical Security Controls® (CIS Controls®)

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

Additional Information

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

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1 - Master Node**

  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 - Master Node**

- **Level 1 - Worker Node**

  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 - Worker Node**

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:

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:

Authors
Rory McCune
Liz Rice

Editor
Randall Mowen

Contributors
Pravin Goyal
Prabhu Angadi
Jordan Liggitt
Eric Chiang
Andrew Peabody
Joe Bowbeer
Mohit Rathore
Jordan Rakoske
Mark Larinde

Recommendations

1 Control Plane Components

This section consists of security recommendations for the direct configuration of Kubernetes control plane processes. These recommendations may not be directly applicable for cluster operators in environments where these components are managed by a 3rd party.

1.1 Control Plane Node Configuration Files

1.1.1 Ensure that the API server pod specification file permissions are set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the API server pod specification file has permissions of `600` 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 by only the administrators on the system.

Impact:

None

Audit:

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

```
stat -c %a /etc/kubernetes/manifests/kube-apiserver.yaml
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/kubernetes/manifests/kube-apiserver.yaml
```

Default Value:

By default, the `kube-apiserver.yaml` file has permissions of `640`.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

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

```
stat -c %U:%G /etc/kubernetes/manifests/kube-apiserver.yaml
```

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

Remediation:

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

```
chown root:root /etc/kubernetes/manifests/kube-apiserver.yaml
```

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

1.1.3 Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the controller manager pod specification file has permissions of `600` 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:

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

```
stat -c %a /etc/kubernetes/manifests/kube-controller-manager.yaml
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/kubernetes/manifests/kube-controller-manager.yaml
```

Default Value:

By default, the `kube-controller-manager.yaml` file has permissions of `640`.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

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

```
stat -c %U:%G /etc/kubernetes/manifests/kube-controller-manager.yaml
```

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

Remediation:

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

```
chown root:root /etc/kubernetes/manifests/kube-controller-manager.yaml
```

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/kube-controller-manager

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

1.1.5 Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the scheduler pod specification file has permissions of `600` 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:

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

```
stat -c %a /etc/kubernetes/manifests/kube-scheduler.yaml
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/kubernetes/manifests/kube-scheduler.yaml
```

Default Value:

By default, `kube-scheduler.yaml` file has permissions of `640`.

References:

1. https://kubernetes.io/docs/admin/kube-scheduler/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

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

```
stat -c %U:%G /etc/kubernetes/manifests/kube-scheduler.yaml
```

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

Remediation:

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

```
chown root:root /etc/kubernetes/manifests/kube-scheduler.yaml
```

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/kube-scheduler/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

1.1.7 Ensure that the etcd pod specification file permissions are set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the `/etc/kubernetes/manifests/etcd.yaml` file has permissions of `600` 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:

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

```
stat -c %a /etc/kubernetes/manifests/etcd.yaml
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/kubernetes/manifests/etcd.yaml
```

Default Value:

By default, `/etc/kubernetes/manifests/etcd.yaml` file has permissions of `640`.

References:

1. https://coreos.com/etcd
2. https://kubernetes.io/docs/admin/etcd/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

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

```
stat -c %U:%G /etc/kubernetes/manifests/etcd.yaml
```

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

Remediation:

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

```
chown root:root /etc/kubernetes/manifests/etcd.yaml
```

Default Value:

By default, `/etc/kubernetes/manifests/etcd.yaml` file ownership is set to `root:root`.

References:

1. https://coreos.com/etcd
2. https://kubernetes.io/docs/admin/etcd/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

1.1.9 Ensure that the Container Network Interface file permissions are set to 600 or more restrictive (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the Container Network Interface files have permissions of `600` 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:

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

```
stat -c %a <path/to/cni/files>
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 <path/to/cni/files>
```

Default Value:

NA

References:

1. https://kubernetes.io/docs/concepts/cluster-administration/networking/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

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

```
stat -c %U:%G <path/to/cni/files>
```

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

Remediation:

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

```
chown root:root <path/to/cni/files>
```

Default Value:

NA

References:

1. https://kubernetes.io/docs/concepts/cluster-administration/networking/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

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

Profile Applicability:

- Level 1 - Master Node

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:

On the etcd server node, get the etcd data directory, passed as an argument `--data-dir`, from the below command:

```
ps -ef | grep etcd
```

Run the below command (based on the etcd data directory found above). For example,

```
stat -c %a /var/lib/etcd
```

Verify that the permissions are `700` or more restrictive.

Remediation:

On the etcd server node, get the etcd data directory, passed as an argument `--data-dir`, from the below command:

```
ps -ef | grep etcd
```

Run the below command (based on the etcd data directory found above). For example,

```
chmod 700 /var/lib/etcd
```

Default Value:

By default, etcd data directory has permissions of `755`.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/configuration.html#data-dir

2. https://kubernetes.io/docs/admin/etcd/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

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 `etcd:etcd`.

Impact:

None

Audit:

On the etcd server node, get the etcd data directory, passed as an argument `--data-dir`, from the below command:

```
ps -ef | grep etcd
```

Run the below command (based on the etcd data directory found above). For example,

```
stat -c %U:%G /var/lib/etcd
```

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

Remediation:

On the etcd server node, get the etcd data directory, passed as an argument `--data-dir`, from the below command:

```
ps -ef | grep etcd
```

Run the below command (based on the etcd data directory found above). For example,

```
chown etcd:etcd /var/lib/etcd
```

Default Value:

By default, etcd data directory ownership is set to `etcd:etcd`.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/configuration.html#data-dir
2. https://kubernetes.io/docs/admin/etcd/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	16 Account Monitoring and Control Account Monitoring and Control			

1.1.13 Ensure that the default administrative credential file permissions are set to 600 (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the `admin.conf` file (and `super-admin.conf` file, where it exists) have permissions of `600`.

Rationale:

As part of initial cluster setup, default kubeconfig files are created to be used by the administrator of the cluster. These files contain private keys and certificates which allow for privileged access to the cluster. You should restrict their file permissions to maintain the integrity and confidentiality of the file(s). The file(s) should be readable and writable by only the administrators on the system.

Impact:

None.

Audit:

Run the following command (based on the file location on your system) on the Control Plane node. For example,

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

On Kubernetes version 1.29 and higher run the following command as well :-

```
stat -c %a /etc/kubernetes/super-admin.conf
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/kubernetes/admin.conf
```

On Kubernetes 1.29+ the `super-admin.conf` file should also be modified, if present. For example,

```
chmod 600 /etc/kubernetes/super-admin.conf
```

Default Value:

By default, admin.conf and super-admin.conf have permissions of `600`.

References:

1. https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/
2. https://raesene.github.io/blog/2024/01/06/when-is-admin-not-admin/

CIS Controls:

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

1.1.14 Ensure that the default administrative credential file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the `admin.conf` (and `super-admin.conf` file, where it exists) file ownership is set to `root:root`.

Rationale:

As part of initial cluster setup, default kubeconfig files are created to be used by the administrator of the cluster. These files contain private keys and certificates which allow for privileged access to the cluster. You should set their file ownership to maintain the integrity and confidentiality of the file. The file(s) should be owned by root:root.

Impact:

None.

Audit:

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

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

On Kubernetes version 1.29 and higher run the following command as well :-

```
stat -c %a /etc/kubernetes/super-admin.conf
```

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

Remediation:

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

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

On Kubernetes 1.29+ the super-admin.conf file should also be modified, if present. For example,

```
chown root:root /etc/kubernetes/super-admin.conf
```

Default Value:

By default, `admin.conf` and `super-admin.conf` file ownership is set to `root:root`.

References:

1. https://kubernetes.io/docs/admin/kubeadm/
2. https://raesene.github.io/blog/2024/01/06/when-is-admin-not-admin/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

1.1.15 Ensure that the scheduler.conf file permissions are set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the `scheduler.conf` file has permissions of `600` 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:

Run the following command (based on the file location on your system) on the Control Plane node. For example,

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

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/kubernetes/scheduler.conf
```

Default Value:

By default, `scheduler.conf` has permissions of `640`.

References:

1. https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

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

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

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

Remediation:

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

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

Default Value:

By default, `scheduler.conf` file ownership is set to `root:root`.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

1.1.17 Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the `controller-manager.conf` file has permissions of 600 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:

Run the following command (based on the file location on your system) on the Control Plane node. For example,

```
stat -c %a /etc/kubernetes/controller-manager.conf
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/kubernetes/controller-manager.conf
```

Default Value:

By default, `controller-manager.conf` has permissions of `640`.

References:

1. https://kubernetes.io/docs/admin/kube-controller-manager/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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 below command (based on the file location on your system) on the Control Plane node. For example,

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

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

Remediation:

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

```
chown root:root /etc/kubernetes/controller-manager.conf
```

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/kube-controller-manager/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

Kubernetes makes use of a number of certificates as part of its operation. You should set 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:

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

```
ls -laR /etc/kubernetes/pki/
```

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

Remediation:

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

```
chown -R root:root /etc/kubernetes/pki/
```

Default Value:

By default, the /etc/kubernetes/pki/ directory 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/admin/kube-apiserver/

CIS Controls:

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

1.1.20 Ensure that the Kubernetes PKI certificate file permissions are set to 600 or more restrictive (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that Kubernetes PKI certificate files have permissions of `600` or more restrictive.

Rationale:

Kubernetes 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 `600` or more restrictive to protect their integrity.

Impact:

None

Audit:

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

```
stat -c '%a' /etc/kubernetes/pki/*.crt
```

Verify that the permissions are `600` or more restrictive.
or

```
ls -l /etc/kubernetes/pki/*.crt
```

Verify -rw------

Remediation:

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

```
chmod -R 600 /etc/kubernetes/pki/*.crt
```

Default Value:

By default, the certificates used by Kubernetes are set to have permissions of `644`

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

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

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that Kubernetes PKI key files have permissions of `600`.

Rationale:

Kubernetes 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:

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

```
stat -c '%a' /etc/kubernetes/pki/*.key
```

Verify that the permissions are `600` or more restrictive.
or

```
ls -l /etc/kubernetes/pki/*.key
```

Verify -rw------

Remediation:

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

```
chmod -R 600 /etc/kubernetes/pki/*.key
```

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/

CIS Controls:

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

1.2 API Server

This section contains recommendations relating to API server configuration flags

1.2.1 Ensure that the --anonymous-auth argument is set to false (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Disable anonymous requests to the API 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 API server. You should rely on authentication to authorize access and disallow 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 will be rejected.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--anonymous-auth` argument is set to `false`.
Alternative Audit
kubectl get pod -nkube-system -lcomponent=kube-apiserver -o=jsonpath='{range .items[*]}{.spec.containers[*]}.command} {"\n"}{end}' | grep '--anonymous-auth' | grep -i false
If the exit code is '1', then the control isn't present / failed

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the below parameter.

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

Default Value:

By default, anonymous access is enabled.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/authentication/#anonymous-requests

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

You will have to configure and use alternate authentication mechanisms such as certificates. Static token based authentication could not be used.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--token-auth-file` argument does not exist.
Alternative Audit Method
kubectl get pod -nkube-system -lcomponent=kube-apiserver -o=jsonpath='{range .items[*]}{.spec.containers[*].command} {"\n"}{end}' | grep '--token-auth-file' | grep -i false
If the exit code is '1', then the control isn't present / failed

Remediation:

Follow the documentation and configure alternate mechanisms for authentication. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and remove the `--token-auth-file=<filename>` parameter.

Default Value:

By default, `--token-auth-file` argument is not set.

References:

1. https://kubernetes.io/docs/admin/authentication/#static-token-file
2. https://kubernetes.io/docs/admin/kube-apiserver/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.6 Securely Manage Enterprise Assets and Software Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.	●	●	●
v7	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 DenyServiceExternalIPs is set (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

This admission controller rejects all net-new usage of the Service field externalIPs.

Rationale:

Most users do not need the ability to set the `externalIPs` field for a `Service` at all, and cluster admins should consider disabling this functionality by enabling the `DenyServiceExternalIPs` admission controller. Clusters that do need to allow this functionality should consider using some custom policy to manage its usage.

Impact:

When enabled, users of the cluster may not create new Services which use externalIPs and may not add new values to externalIPs on existing Service objects.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `DenyServiceExternalIPs' argument exist as a string value in --disable-admission-plugins.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and remove the `--DenyServiceExternalIPs'parameter
or
The Kubernetes API server flag disable-admission-plugins takes a comma-delimited list of admission control plugins to be disabled, even if they are in the list of plugins enabled by default.

```
kube-apiserver --disable-admission-plugins=DenyServiceExternalIPs,AlwaysDeny
...
```

Default Value:

By default, --disable-admission-plugins=DenyServiceExternalIP argument is not set.

References:

1. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/
2. https://kubernetes.io/docs/admin/kube-apiserver/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.6 Securely Manage Enterprise Assets and Software Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.	●	●	●
v7	16.4 Encrypt or Hash all Authentication Credentials Encrypt or hash with a salt all authentication credentials when stored.		●	●

1.2.4 Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Automated)

Profile Applicability:

- Level 1 - Master Node

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:

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

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--kubelet-client-certificate` and `--kubelet-client-key` arguments exist and they are set as appropriate.
Alternative Audit
kubectl get pod -nkube-system -lcomponent=kube-apiserver -o=jsonpath='{range .items[]}{.spec.containers[].command} {"\n"}{end}' | grep '--kubelet-client-certificate' | grep -i false
If the exit code is '1', then the control isn't present / failed

Remediation:

Follow the Kubernetes documentation and set up the TLS connection between the apiserver and kubelets. Then, edit API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the kubelet client certificate and key parameters as below.

```
--kubelet-client-certificate=<path/to/client-certificate-file>
--kubelet-client-key=<path/to/client-key-file>
```

Default Value:

By default, certificate-based kubelet authentication is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/kubelet-authentication-authorization/
3. https://kubernetes.io/docs/concepts/cluster-administration/master-node-communication/#apiserver---kubelet

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.11 Leverage Vetted Modules or Services for Application Security Components Leverage vetted modules or services for application security components, such as identity management, encryption, and auditing and logging. Using platform features in critical security functions will reduce developers' workload and minimize the likelihood of design or implementation errors. Modern operating systems provide effective mechanisms for identification, authentication, and authorization and make those mechanisms available to applications. Use only standardized, currently accepted, and extensively reviewed encryption algorithms. Operating systems also provide mechanisms to create and maintain secure audit logs.		●	●
v7	9 Limitation and Control of Network Ports, Protocols, and Services Limitation and Control of Network Ports, Protocols, and Services			

1.2.5 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Automated)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--kubelet-certificate-authority` argument exists and is set as appropriate.
Alternative Audit
kubectl get pod -nkube-system -lcomponent=kube-apiserver -o=jsonpath='{range .items[]}{.spec.containers[].command} {"\n"}{end}' | grep '--kubelet-certificate-Authority' | grep -i false
If the exit code is '1', then the control isn't present / failed

Remediation:

Follow the Kubernetes documentation and setup the TLS connection between the apiserver and kubelets. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--kubelet-certificate-authority` parameter to the path to the cert file for the certificate authority.

```
--kubelet-certificate-authority=<ca-string>
```

Default Value:

By default, `--kubelet-certificate-authority` argument is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/kubelet-authentication-authorization/
3. https://kubernetes.io/docs/concepts/cluster-administration/master-node-communication/#apiserver---kubelet

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.11 Leverage Vetted Modules or Services for Application Security Components Leverage vetted modules or services for application security components, such as identity management, encryption, and auditing and logging. Using platform features in critical security functions will reduce developers' workload and minimize the likelihood of design or implementation errors. Modern operating systems provide effective mechanisms for identification, authentication, and authorization and make those mechanisms available to applications. Use only standardized, currently accepted, and extensively reviewed encryption algorithms. Operating systems also provide mechanisms to create and maintain secure audit logs.		●	●
v7	9 Limitation and Control of Network Ports, Protocols, and Services Limitation and Control of Network Ports, Protocols, and Services			

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--authorization-mode` argument exists and is not set to `AlwaysAllow`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--authorization-mode` parameter to values other than `AlwaysAllow`. One such example could be as below.

```
--authorization-mode=RBAC
```

Default Value:

By default, `AlwaysAllow` is not enabled.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/authorization/

CIS Controls:

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

1.2.7 Ensure that the --authorization-mode argument includes Node (Automated)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--authorization-mode` argument exists and is set to a value to include `Node`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--authorization-mode` parameter to a value that includes `Node`.

```
--authorization-mode=Node,RBAC
```

Default Value:

By default, `Node` authorization is not enabled.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/authorization/node/
3. https://github.com/kubernetes/kubernetes/pull/46076
4. https://acotten.com/post/kube17-security

CIS Controls:

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

1.2.8 Ensure that the --authorization-mode argument includes RBAC (Automated)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--authorization-mode` argument exists and is set to a value to include `RBAC`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--authorization-mode` parameter to a value that includes `RBAC`, for example:

```
--authorization-mode=Node,RBAC
```

Default Value:

By default, `RBAC` authorization is not enabled.

References:

1. https://kubernetes.io/docs/reference/access-authn-authz/rbac/

CIS Controls:

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

1.2.9 Ensure that the admission control plugin EventRateLimit is set (Manual)

Profile Applicability:

- Level 1 - Master Node

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.

Impact:

You need to carefully tune in limits as per your environment.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--enable-admission-plugins` argument is set to a value that includes `EventRateLimit`.

Remediation:

Follow the Kubernetes documentation and set the desired limits in a configuration file. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` and set the below parameters.

```
--enable-admission-plugins=...,EventRateLimit,...
--admission-control-config-file=<path/to/configuration/file>
```

Default Value:

By default, `EventRateLimit` is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers/#eventratelimit

3. https://github.com/staebler/community/blob/9873b632f4d99b5d99c38c9b15fe2f8b93d0a746/contributors/design-proposals/admission_control_event_rate_limit.md

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.11 Leverage Vetted Modules or Services for Application Security Components Leverage vetted modules or services for application security components, such as identity management, encryption, and auditing and logging. Using platform features in critical security functions will reduce developers' workload and minimize the likelihood of design or implementation errors. Modern operating systems provide effective mechanisms for identification, authentication, and authorization and make those mechanisms available to applications. Use only standardized, currently accepted, and extensively reviewed encryption algorithms. Operating systems also provide mechanisms to create and maintain secure audit logs.		●	●
v7	8.3 Enable Operating System Anti-Exploitation Features/ Deploy Anti-Exploit Technologies Enable anti-exploitation features such as Data Execution Prevention (DEP) or Address Space Layout Randomization (ASLR) that are available in an operating system or deploy appropriate toolkits that can be configured to apply protection to a broader set of applications and executables.		●	●

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

Profile Applicability:

- Level 1 - Master Node

Description:

Do not allow all requests.

Rationale:

Setting admission control plugin `AlwaysAdmit` allows all requests and do 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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that if the `--enable-admission-plugins` argument is set, its value does not include `AlwaysAdmit`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and either remove the `--enable-admission-plugins` parameter, or set it to a value that does not include `AlwaysAdmit`.

Default Value:

`AlwaysAdmit` is not in the list of default admission plugins.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers/#alwaysadmit

CIS Controls:

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

1.2.11 Ensure that the admission control plugin AlwaysPullImages is set (Manual)

Profile Applicability:

- Level 1 - Master Node

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.

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 on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--enable-admission-plugins` argument is set to a value that includes `AlwaysPullImages`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--enable-admission-plugins` parameter to include `AlwaysPullImages`.

```
--enable-admission-plugins=...,AlwaysPullImages,...
```

Default Value:

By default, `AlwaysPullImages` is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers/#alwayspullimages

CIS Controls:

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

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

Profile Applicability:

- Level 2 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--disable-admission-plugins` argument is set to a value that does not includes `ServiceAccount`.

Remediation:

Follow the documentation and create `ServiceAccount` objects as per your environment. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and ensure that the `--disable-admission-plugins` parameter is set to a value that does not include `ServiceAccount`.

Default Value:

By default, `ServiceAccount` is set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers/#serviceaccount
3. https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

CIS Controls:

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

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

Profile Applicability:

- Level 2 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--disable-admission-plugins` argument is set to a value that does not include `NamespaceLifecycle`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--disable-admission-plugins` parameter to ensure it does not include `NamespaceLifecycle`.

Default Value:

By default, `NamespaceLifecycle` is set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers/#namespacelifecycle

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4 Secure Configuration of Enterprise Assets and Software Establish and maintain the secure configuration of enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/IoT devices; and servers) and software (operating systems and applications).			
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

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

Profile Applicability:

- Level 2 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--enable-admission-plugins` argument is set to a value that includes `NodeRestriction`.

Remediation:

Follow the Kubernetes documentation and configure `NodeRestriction` plug-in on kubelets. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--enable-admission-plugins` parameter to a value that includes `NodeRestriction`.

```
--enable-admission-plugins=...,NodeRestriction,...
```

Default Value:

By default, `NodeRestriction` is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers/#noderestriction
3. https://kubernetes.io/docs/admin/authorization/node/
4. https://acotten.com/post/kube17-security

CIS Controls:

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

1.2.15 Ensure that the --profiling argument is set to false (Automated)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--profiling` argument is set to `false`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the below parameter.

```
--profiling=false
```

Default Value:

By default, profiling is enabled.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://github.com/kubernetes/community/blob/master/contributors/devel/profiling.md

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8 Audit Log Management Collect, alert, review, and retain audit logs of events that could help detect, understand, or recover from an attack.			
v7	6 Maintenance, Monitoring and Analysis of Audit Logs Maintenance, Monitoring and Analysis of Audit Logs			

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

Profile Applicability:

- Level 1 - Master Node

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 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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--audit-log-path` argument is set as appropriate.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--audit-log-path` parameter to a suitable path and file where you would like audit logs to be written, for example:

```
--audit-log-path=/var/log/apiserver/audit.log
```

Default Value:

By default, auditing is not enabled.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/concepts/cluster-administration/audit/
3. https://github.com/kubernetes/features/issues/22

CIS Controls:

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

1.2.17 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Automated)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--audit-log-maxage` argument is set to `30` or as appropriate.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--audit-log-maxage` parameter to 30 or as an appropriate number of days:

```
--audit-log-maxage=30
```

Default Value:

By default, auditing is not enabled.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/concepts/cluster-administration/audit/
3. https://github.com/kubernetes/features/issues/22

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.3 Ensure Adequate Audit Log Storage Ensure that logging destinations maintain adequate storage to comply with the enterprise's audit log management process.	●	●	●
v7	6.4 Ensure adequate storage for logs Ensure that all systems that store logs have adequate storage space for the logs generated.		●	●

1.2.18 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Automated)

Profile Applicability:

- Level 1 - Master Node

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 approximate have 1 GB of log data that you could potentially use for your analysis.

Impact:

None

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--audit-log-maxbackup` argument is set to `10` or as appropriate.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--audit-log-maxbackup` parameter to 10 or to an appropriate value.

```
--audit-log-maxbackup=10
```

Default Value:

By default, auditing is not enabled.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/concepts/cluster-administration/audit/
3. https://github.com/kubernetes/features/issues/22

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.3 Ensure Adequate Audit Log Storage Ensure that logging destinations maintain adequate storage to comply with the enterprise's audit log management process.	●	●	●
v7	6.4 Ensure adequate storage for logs Ensure that all systems that store logs have adequate storage space for the logs generated.		●	●

1.2.19 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Automated)

Profile Applicability:

- Level 1 - Master Node

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 approximate have 1 GB of log data that you could potentially use for your analysis.

Impact:

None

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--audit-log-maxsize` argument is set to `100` or as appropriate.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--audit-log-maxsize` parameter to an appropriate size in MB. For example, to set it as 100 MB:

```
--audit-log-maxsize=100
```

Default Value:

By default, auditing is not enabled.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/concepts/cluster-administration/audit/
3. https://github.com/kubernetes/features/issues/22

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.3 Ensure Adequate Audit Log Storage Ensure that logging destinations maintain adequate storage to comply with the enterprise's audit log management process.	●	●	●
v7	6.4 Ensure adequate storage for logs Ensure that all systems that store logs have adequate storage space for the logs generated.		●	●

1.2.20 Ensure that the --request-timeout argument is set as appropriate (Manual)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--request-timeout` argument is either not set or set to an appropriate value.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` and set the below parameter as appropriate and if needed. For example,

```
--request-timeout=300s
```

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://github.com/kubernetes/kubernetes/pull/51415

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4 Secure Configuration of Enterprise Assets and Software Establish and maintain the secure configuration of enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/IoT devices; and servers) and software (operating systems and applications).			
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that if the `--service-account-lookup` argument exists it is set to `true`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the below parameter.

```
--service-account-lookup=true
```

Alternatively, you can delete the `--service-account-lookup` parameter from this file so that the default takes effect.

Default Value:

By default, `--service-account-lookup` argument is set to `true`.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://github.com/kubernetes/kubernetes/issues/24167
3. https://en.wikipedia.org/wiki/Time_of_check_to_time_of_use

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	16 Account Monitoring and Control Account Monitoring and Control			

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--service-account-key-file` argument exists and is set as appropriate.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the Control Plane node and set the `--service-account-key-file` parameter to the public key file for service accounts:

```
--service-account-key-file=<filename>
```

Default Value:

By default, `--service-account-key-file` argument is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://github.com/kubernetes/kubernetes/issues/24167

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

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

Profile Applicability:

- Level 1 - Master Node

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 must be configured for etcd.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--etcd-certfile` and `--etcd-keyfile` arguments exist and they are set as appropriate.

Remediation:

Follow the Kubernetes documentation and set up the TLS connection between the apiserver and etcd. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the etcd certificate and key file parameters.

```
--etcd-certfile=<path/to/client-certificate-file>
--etcd-keyfile=<path/to/client-key-file>
```

Default Value:

By default, `--etcd-certfile` and `--etcd-keyfile` arguments are not set

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://coreos.com/etcd/docs/latest/op-guide/security.html

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--tls-cert-file` and `--tls-private-key-file` arguments exist and they are set as appropriate.

Remediation:

Follow the Kubernetes documentation and set up the TLS connection on the apiserver. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the TLS certificate and private key file parameters.

```
--tls-cert-file=<path/to/tls-certificate-file>
--tls-private-key-file=<path/to/tls-key-file>
```

Default Value:

By default, `--tls-cert-file` and `--tls-private-key-file` are presented and created for use.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. http://rootsquash.com/2016/05/10/securing-the-kubernetes-api/
3. https://github.com/kelseyhightower/docker-kubernetes-tls-guide

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--client-ca-file` argument exists and it is set as appropriate.

Remediation:

Follow the Kubernetes documentation and set up the TLS connection on the apiserver. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the client certificate authority file.

```
--client-ca-file=<path/to/client-ca-file>
```

Default Value:

By default, `--client-ca-file` argument is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. http://rootsquash.com/2016/05/10/securing-the-kubernetes-api/
3. https://github.com/kelseyhightower/docker-kubernetes-tls-guide

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--etcd-cafile` argument exists and it is set as appropriate.

Remediation:

Follow the Kubernetes documentation and set up the TLS connection between the apiserver and etcd. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the etcd certificate authority file parameter.

```
--etcd-cafile=<path/to/ca-file>
```

Default Value:

By default, `--etcd-cafile` is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://coreos.com/etcd/docs/latest/op-guide/security.html

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

None

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--encryption-provider-config` argument is set to a `EncryptionConfig` file. Additionally, ensure that the `EncryptionConfig` file has all the desired `resources` covered especially any secrets.

Remediation:

Follow the Kubernetes documentation and configure a `EncryptionConfig` file. Then, edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--encryption-provider-config` parameter to the path of that file:

```
--encryption-provider-config=</path/to/EncryptionConfig/File>
```

Default Value:

By default, `--encryption-provider-config` is not set.

References:

1. https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
2. https://acotten.com/post/kube17-security
3. https://kubernetes.io/docs/admin/kube-apiserver/
4. https://github.com/kubernetes/features/issues/92

CIS Controls:

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

1.2.28 Ensure that encryption providers are appropriately configured (Manual)

Profile Applicability:

- Level 1 - Master Node

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:

None

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Get the `EncryptionConfig` file set for `--encryption-provider-config` argument. Verify that `aescbc`, `kms` or `secretbox` is set as the encryption provider for all the desired `resources`.

Remediation:

Follow the Kubernetes documentation and configure a `EncryptionConfig` file. In this file, choose `aescbc`, `kms` or `secretbox` as the encryption provider.

Default Value:

By default, no encryption provider is set.

References:

1. https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
2. https://acotten.com/post/kube17-security
3. https://kubernetes.io/docs/admin/kube-apiserver/
4. https://github.com/kubernetes/features/issues/92
5. https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/#providers

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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 modern cryptographic ciphers will not be able to make connections to the API server.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-apiserver
```

Verify that the `--tls-cipher-suites` argument is set as outlined in the remediation procedure below.

Remediation:

Edit the API server pod specification file /etc/kubernetes/manifests/kube-apiserver.yaml on the Control Plane node and set the below parameter.

```
--tls-cipher-
suites=TLS_AES_128_GCM_SHA256,TLS_AES_256_GCM_SHA384,TLS_CHACHA20_POLY1305_SH
A256,TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SH
A256,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SH
A384,TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305,TLS_ECDHE_ECDSA_WITH_CHACHA20_POL
Y1305_SHA256,TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_128_C
BC_SHA,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_CBC_S
HA,TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
,TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256,TLS_RSA_WITH_3DES_EDE_CBC_SHA,TL
S_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_2
56_CBC_SHA, TLS_RSA_WITH_AES_256_GCM_SHA384.
```

Default Value:

By default the Kubernetes API server supports a wide range of TLS ciphers

References:

1. https://github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices#23-use-secure-cipher-suites

Additional Information:

The list chosen above should be fine for modern clients. It's essentially the list from the Mozilla "Modern cipher" option with the ciphersuites supporting CBC mode removed, as CBC has traditionally had a lot of issues

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

1.3 Controller Manager

This section contains recommendations relating to Controller Manager configuration flags

1.3.1 Ensure that the --terminated-pod-gc-threshold argument is set as appropriate (Manual)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-controller-manager
```

Verify that the `--terminated-pod-gc-threshold` argument is set as appropriate.

Remediation:

Edit the Controller Manager pod specification file `/etc/kubernetes/manifests/kube-controller-manager.yaml` on the Control Plane node and set the `--terminated-pod-gc-threshold` to an appropriate threshold, for example:

```
--terminated-pod-gc-threshold=10
```

Default Value:

By default, `--terminated-pod-gc-threshold` is set to `12500`.

References:

1. https://kubernetes.io/docs/admin/kube-controller-manager/
2. https://github.com/kubernetes/kubernetes/issues/28484

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	10.5 Enable Anti-Exploitation Features Enable anti-exploitation features on enterprise assets and software, where possible, such as Microsoft® Data Execution Prevention (DEP), Windows® Defender Exploit Guard (WDEG), or Apple® System Integrity Protection (SIP) and Gatekeeper™.		●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

1.3.2 Ensure that the --profiling argument is set to false (Automated)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-controller-manager
```

Verify that the `--profiling` argument is set to `false`.

Remediation:

Edit the Controller Manager pod specification file `/etc/kubernetes/manifests/kube-controller-manager.yaml` on the Control Plane node and set the below parameter.

```
--profiling=false
```

Default Value:

By default, profiling is enabled.

References:

1. https://kubernetes.io/docs/admin/kube-controller-manager/
2. https://github.com/kubernetes/community/blob/master/contributors/devel/profiling.md

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-controller-manager
```

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

Remediation:

Edit the Controller Manager pod specification file `/etc/kubernetes/manifests/kube-controller-manager.yaml` on the Control Plane node to set the below parameter.

```
--use-service-account-credentials=true
```

Default Value:

By default, `--use-service-account-credentials` is set to false.

References:

1. https://kubernetes.io/docs/admin/kube-controller-manager/
2. https://kubernetes.io/docs/admin/service-accounts-admin/
3. https://github.com/kubernetes/kubernetes/blob/release-1.6/plugin/pkg/auth/authorizer/rbac/bootstrappolicy/testdata/controller-roles.yaml
4. https://github.com/kubernetes/kubernetes/blob/release-1.6/plugin/pkg/auth/authorizer/rbac/bootstrappolicy/testdata/controller-role-bindings.yaml
5. https://kubernetes.io/docs/admin/authorization/rbac/#controller-roles

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-controller-manager
```

Verify that the `--service-account-private-key-file` argument is set as appropriate.

Remediation:

Edit the Controller Manager pod specification file `/etc/kubernetes/manifests/kube-controller-manager.yaml` on the Control Plane node and set the `--service-account-private-key-file` parameter to the private key file for service accounts.

```
--service-account-private-key-file=<filename>
```

Default Value:

By default, `--service-account-private-key-file` it not set.

References:

1. https://kubernetes.io/docs/admin/kube-controller-manager/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●
v6	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

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

Profile Applicability:

- Level 1 - Master Node

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:

You need to setup and maintain root certificate authority file.

Audit:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-controller-manager
```

Verify that the `--root-ca-file` argument exists and is set to a certificate bundle file containing the root certificate for the API server's serving certificate.

Remediation:

Edit the Controller Manager pod specification file `/etc/kubernetes/manifests/kube-controller-manager.yaml` on the Control Plane node and set the `--root-ca-file` parameter to the certificate bundle file`.

```
--root-ca-file=<path/to/file>
```

Default Value:

By default, `--root-ca-file` is not set.

References:

1. https://kubernetes.io/docs/admin/kube-controller-manager/
2. https://github.com/kubernetes/kubernetes/issues/11000

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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 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 the Control Plane node:

```
ps -ef | grep kube-controller-manager
```

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

Remediation:

Edit the Controller Manager pod specification file `/etc/kubernetes/manifests/kube-controller-manager.yaml` on the Control Plane node and set the `--feature-gates` parameter to include `RotateKubeletServerCertificate=true`.

```
--feature-gates=RotateKubeletServerCertificate=true
```

Default Value:

By default, `RotateKubeletServerCertificate` is set to "true" this recommendation verifies that it has not been disabled.

References:

1. https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/#approval-controller

2. https://github.com/kubernetes/features/issues/267
3. https://github.com/kubernetes/kubernetes/pull/45059
4. https://kubernetes.io/docs/admin/kube-controller-manager/

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

The Controller Manager API service which runs on port 10252/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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-controller-manager
```

Verify that the `--bind-address` argument is set to 127.0.0.1

Remediation:

Edit the Controller Manager pod specification file `/etc/kubernetes/manifests/kube-controller-manager.yaml` on the Control Plane node and ensure the correct value for the `--bind-address` parameter

Default Value:

By default, the `--bind-address` parameter is set to 0.0.0.0

References:

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

Additional Information:

Although the current Kubernetes documentation site says that `--address` is deprecated in favour of `--bind-address` Kubeadm 1.11 still makes use of `--address`

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.6 Use of Secure Network Management and Communication Protocols Use secure network management and communication protocols (e.g., 802.1X, Wi-Fi Protected Access 2 (WPA2) Enterprise or greater).		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

1.4 Scheduler

This section contains recommendations relating to Scheduler configuration flags

1.4.1 Ensure that the --profiling argument is set to false (Automated)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-scheduler
```

Verify that the `--profiling` argument is set to `false`.

Remediation:

Edit the Scheduler pod specification file `/etc/kubernetes/manifests/kube-scheduler.yaml` file on the Control Plane node and set the below parameter.

```
--profiling=false
```

Default Value:

By default, profiling is enabled.

References:

1. https://kubernetes.io/docs/admin/kube-scheduler/
2. https://github.com/kubernetes/community/blob/master/contributors/devel/profiling.md

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the Control Plane node:

```
ps -ef | grep kube-scheduler
```

Verify that the `--bind-address` argument is set to 127.0.0.1

Remediation:

Edit the Scheduler pod specification file `/etc/kubernetes/manifests/kube-scheduler.yaml` on the Control Plane node and ensure the correct value for the `--bind-address` parameter

Default Value:

By default, the `--bind-address` parameter is set to 0.0.0.0

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.6 Use of Secure Network Management and Communication Protocols Use secure network management and communication protocols (e.g., 802.1X, Wi-Fi Protected Access 2 (WPA2) Enterprise or greater).		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

2 etcd

This section covers recommendations for etcd configuration.

This sections assumes you're running etcd in a Kubernetes pod. If you are running etcd externally the file paths, audit and remediation process my vary.

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the etcd server node

```
ps -ef | grep etcd
```

Verify that the `--cert-file` and the `--key-file` arguments are set as appropriate.

Remediation:

Follow the etcd service documentation and configure TLS encryption.
Then, edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and set the below parameters.

```
--cert-file=</path/to/ca-file>
--key-file=</path/to/key-file>
```

Default Value:

By default, TLS encryption is not set.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/security.html
2. https://kubernetes.io/docs/admin/etcd/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.		●	●
v7	14.4 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 - Master Node

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:

Run the following command on the etcd server node:

```
ps -ef | grep etcd
```

Verify that the `--client-cert-auth` argument is set to `true`.

Remediation:

Edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and set the below parameter.

```
--client-cert-auth="true"
```

Default Value:

By default, the etcd service can be queried by unauthenticated clients.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/security.html
2. https://kubernetes.io/docs/admin/etcd/
3. https://coreos.com/etcd/docs/latest/op-guide/configuration.html#client-cert-auth

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on the etcd server node:

```
ps -ef | grep etcd
```

Verify that if the `--auto-tls` argument exists, it is not set to `true`.

Remediation:

Edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and either remove the `--auto-tls` parameter or set it to `false`.

```
--auto-tls=false
```

Default Value:

By default, `--auto-tls` is set to `false`.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/security.html
2. https://kubernetes.io/docs/admin/etcd/
3. https://coreos.com/etcd/docs/latest/op-guide/configuration.html#auto-tls

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.		●	●
v7	14.4 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 - Master Node

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 would need to set up TLS for their communication.

Audit:

Run the following command on the etcd server node:

```
ps -ef | grep etcd
```

Verify that the `--peer-cert-file` and `--peer-key-file` arguments are set as appropriate.
Note: This recommendation is applicable only for etcd clusters. If you are using only one etcd server in your environment then this recommendation is not applicable.

Remediation:

Follow the etcd service documentation and configure peer TLS encryption as appropriate for your etcd cluster.
Then, edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and set the below parameters.

```
--peer-client-file=</path/to/peer-cert-file>
--peer-key-file=</path/to/peer-key-file>
```

Default Value:

Note: This recommendation is applicable only for etcd clusters. If you are using only one etcd server in your environment then this recommendation is not applicable.

By default, peer communication over TLS is not configured.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/security.html
2. https://kubernetes.io/docs/admin/etcd/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.		●	●
v7	14.4 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 - Master Node

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 will require a valid client certificate for authentication.

Audit:

Run the following command on the etcd server node:

```
ps -ef | grep etcd
```

Verify that the `--peer-client-cert-auth` argument is set to `true`.
Note: This recommendation is applicable only for etcd clusters. If you are using only one etcd server in your environment then this recommendation is not applicable.

Remediation:

Edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and set the below parameter.

```
--peer-client-cert-auth=true
```

Default Value:

Note: This recommendation is applicable only for etcd clusters. If you are using only one etcd server in your environment then this recommendation is not applicable.

By default, `--peer-client-cert-auth` argument is set to `false`.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/security.html
2. https://kubernetes.io/docs/admin/etcd/

3. https://coreos.com/etcd/docs/latest/op-guide/configuration.html#peer-client-cert-auth

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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 will require a valid client certificate for authentication.

Audit:

Run the following command on the etcd server node:

```
ps -ef | grep etcd
```

Verify that if the `--peer-auto-tls` argument exists, it is not set to `true`.
Note: This recommendation is applicable only for etcd clusters. If you are using only one etcd server in your environment then this recommendation is not applicable.

Remediation:

Edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and either remove the `--peer-auto-tls` parameter or set it to `false`.

```
--peer-auto-tls=false
```

Default Value:

Note: This recommendation is applicable only for etcd clusters. If you are using only one etcd server in your environment then this recommendation is not applicable.

By default, `--peer-auto-tls` argument is set to `false`.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/security.html

2. https://kubernetes.io/docs/admin/etcd/
3. https://coreos.com/etcd/docs/latest/op-guide/configuration.html#peer-auto-tls

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.	●	●	●
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.		●	●

2.7 Ensure that a unique Certificate Authority is used for etcd (Manual)

Profile Applicability:

- Level 2 - Master Node

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:

Review the CA used by the etcd environment and ensure that it does not match the CA certificate file used for the management of the overall Kubernetes cluster.
Run the following command on the master node:

```
ps -ef | grep etcd
```

Note the file referenced by the `--trusted-ca-file` argument.
Run the following command on the master node:

```
ps -ef | grep apiserver
```

Verify that the file referenced by the `--client-ca-file` for apiserver is different from the `--trusted-ca-file` used by etcd.

Remediation:

Follow the etcd documentation and create a dedicated certificate authority setup for the etcd service.
Then, edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and set the below parameter.

```
--trusted-ca-file=</path/to/ca-file>
```

Default Value:

By default, no etcd certificate is created and used.

References:

1. https://coreos.com/etcd/docs/latest/op-guide/security.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.1 Establish an Access Granting Process Establish and follow a process, preferably automated, for granting access to enterprise assets upon new hire, rights grant, or role change of a user.	●	●	●
v8	6.2 Establish an Access Revoking Process Establish and follow a process, preferably automated, for revoking access to enterprise assets, through disabling accounts immediately upon termination, rights revocation, or role change of a user. Disabling accounts, instead of deleting accounts, may be necessary to preserve audit trails.	●	●	●

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 1 - Master Node

Description:

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

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

Rationale:

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

Impact:

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

Audit:

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

Remediation:

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

Default Value:

Client certificate authentication is enabled by default.

Additional Information:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.2 Establish an Access Revoking Process Establish and follow a process, preferably automated, for revoking access to enterprise assets, through disabling accounts immediately upon termination, rights revocation, or role change of a user. Disabling accounts, instead of deleting accounts, may be necessary to preserve audit trails.	●	●	●
v7	16.7 Establish Process for Revoking Access Establish and follow an automated process for revoking system access by disabling accounts immediately upon termination or change of responsibilities of an employee or contractor . Disabling these accounts, instead of deleting accounts, allows preservation of audit trails.		●	●

3.1.2 Service account token authentication should not be used for users (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Kubernetes provides service account tokens which are intended for use by workloads running in the Kubernetes cluster, for authentication to the API server.

These tokens are not designed for use by end-users and do not provide for features such as revocation or expiry, making them insecure. A newer version of the feature (Bound service account token volumes) does introduce expiry but still does not allow for specific revocation.

Rationale:

With any authentication mechanism the ability to revoke credentials if they are compromised or no longer required, is a key control. Service account token authentication does not allow for this due to the use of JWT tokens as an underlying technology.

Impact:

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

Audit:

Review user access to the cluster and ensure that users are not making use of service account token authentication.

Remediation:

Alternative mechanisms provided by Kubernetes such as the use of OIDC should be implemented in place of service account tokens.

Default Value:

Service account token authentication is enabled by default.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.2 Establish an Access Revoking Process Establish and follow a process, preferably automated, for revoking access to enterprise assets, through disabling accounts immediately upon termination, rights revocation, or role change of a user. Disabling accounts, instead of deleting accounts, may be necessary to preserve audit trails.	●	●	●
v7	16.7 Establish Process for Revoking Access Establish and follow an automated process for revoking system access by disabling accounts immediately upon termination or change of responsibilities of an employee or contractor . Disabling these accounts, instead of deleting accounts, allows preservation of audit trails.		●	●

3.1.3 Bootstrap token authentication should not be used for users (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Kubernetes provides bootstrap tokens which are intended for use by new nodes joining the cluster

These tokens are not designed for use by end-users they are specifically designed for the purpose of bootstrapping new nodes and not for general authentication

Rationale:

Bootstrap tokens are not intended for use as a general authentication mechanism and impose constraints on user and group naming that do not facilitate good RBAC design. They also cannot be used with MFA resulting in a weak authentication mechanism being available.

Impact:

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

Audit:

Review user access to the cluster and ensure that users are not making use of bootstrap token authentication.

Remediation:

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

Default Value:

Bootstrap token authentication is not enabled by default and requires an API server parameter to be set.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.2 Establish an Access Revoking Process Establish and follow a process, preferably automated, for revoking access to enterprise assets, through disabling accounts immediately upon termination, rights revocation, or role change of a user. Disabling accounts, instead of deleting accounts, may be necessary to preserve audit trails.	●	●	●
v7	16.7 Establish Process for Revoking Access Establish and follow an automated process for revoking system access by disabling accounts immediately upon termination or change of responsibilities of an employee or contractor . Disabling these accounts, instead of deleting accounts, allows preservation of audit trails.		●	●

3.2 Logging

3.2.1 Ensure that a minimal audit policy is created (Manual)

Profile Applicability:

- Level 1 - Master Node

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:

Run the following command on one of the cluster master nodes:

```
ps -ef | grep kube-apiserver
```

Verify that the `--audit-policy-file` is set. Review the contents of the file specified and ensure that it contains a valid audit policy.

Remediation:

Create an audit policy file for your cluster.

Default Value:

Unless the `--audit-policy-file` flag is specified, no auditing will be carried out.

References:

1. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.	●	●	●

Controls Version	Control	IG 1	IG 2	IG 3
v7	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 - Master Node

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 destination.

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).

Remediation:

Consider modification of the audit policy in use on the cluster to include these items, at a minimum.

Default Value:

By default Kubernetes clusters do not log audit information.

References:

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

3. https://github.com/falcosecurity/falco/blob/master/examples/k8s_audit_config/audit-policy.yaml
4. https://github.com/kubernetes/kubernetes/blob/master/cluster/gce/gci/configure-helper.sh#L735

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.5 Collect Detailed Audit Logs Configure detailed audit logging for enterprise assets containing sensitive data. Include event source, date, username, timestamp, source addresses, destination addresses, and other useful elements that could assist in a forensic investigation.		●	●
v7	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.

To Perform an Automated Audit utilizing CIS-CAT the following parameters must be set on each node being evaluated.

`$kubelet_service_config`

`$kubelet_config`

`$kubelet_config_yaml`

If you are auditing a kubeadm environment the default settings for these values are below:

```
export kubelet_service_config=/etc/systemd/system/kubelet.service.d/10-
kubeadm.conf
export kubelet_config=/etc/kubernetes/kubelet.conf
export kubelet_config_yaml=/var/lib/kubelet/config.yaml
```

4.1.1 Ensure that the kubelet service file permissions are set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the `kubelet` service file has permissions of `600` 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:

Automated AAC auditing has been modified to allow CIS-CAT to input a variable for the <PATH>/<FILENAME> of the kubelet service config file.
Please set $kubelet_service_config=<PATH> based on the file location on your system for example:

```
export
kubelet_service_config=/etc/systemd/system/kubelet.service.d/kubeadm.conf
```

To perform the audit manually:
Run the below command (based on the file location on your system) on the each worker node. For example,

```
stat -c %a /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/systemd/system/kubelet.service.d/kubeadm.conf
```

Default Value:

By default, the `kubelet` service file has permissions of `640`.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/#44-joining-your-nodes
3. https://kubernetes.io/docs/admin/kubeadm/#kubelet-drop-in

CIS Controls:

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

4.1.2 Ensure that the kubelet service file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1 - Worker Node

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:

Automated AAC auditing has been modified to allow CIS-CAT to input a variable for the <PATH>/<FILENAME> of the kubelet service config file.
Please set $kubelet_service_config=<PATH> based on the file location on your system for example:

```
export
kubelet_service_config=/etc/systemd/system/kubelet.service.d/kubeadm.conf
```

To perform the audit manually:
Run the below command (based on the file location on your system) on the each worker node. For example,

```
stat -c %U:%G /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

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

Remediation:

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

```
chown root:root /etc/systemd/system/kubelet.service.d/kubeadm.conf
```

Default Value:

By default, `kubelet` service file ownership is set to `root:root`.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/#44-joining-your-nodes
3. https://kubernetes.io/docs/admin/kubeadm/#kubelet-drop-in

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

4.1.3 If proxy kubeconfig file exists ensure permissions are set to 600 or more restrictive (Manual)

Profile Applicability:

- Level 1 - Worker Node

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 `600` 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:

Find the kubeconfig file being used by `kube-proxy` by running the following command:

```
ps -ef | grep kube-proxy
```

If `kube-proxy` is running, get the kubeconfig file location from the `--kubeconfig` parameter.
To perform the audit:
Run the below command (based on the file location on your system) on the each worker node. For example,

```
stat -c %a <path><filename>
```

Verify that a file is specified and it exists with permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 <proxy kubeconfig file>
```

Default Value:

By default, proxy file has permissions of `640`.

References:

1. https://kubernetes.io/docs/admin/kube-proxy/

CIS Controls:

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

4.1.4 If proxy kubeconfig file exists ensure ownership is set to root:root (Manual)

Profile Applicability:

- Level 1 - Worker Node

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:

Find the kubeconfig file being used by `kube-proxy` by running the following command:

```
ps -ef | grep kube-proxy
```

If `kube-proxy` is running, get the kubeconfig file location from the `--kubeconfig` parameter.
To perform the audit:
Run the below command (based on the file location on your system) on the each worker node. For example,

```
stat -c %U:%G <path><filename>
```

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

Remediation:

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

```
chown root:root <proxy kubeconfig file>
```

Default Value:

By default, `proxy` file ownership is set to `root:root`.

References:

1. https://kubernetes.io/docs/admin/kube-proxy/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

4.1.5 Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Worker Node

Description:

Ensure that the `kubelet.conf` file has permissions of `600` 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:

Automated AAC auditing has been modified to allow CIS-CAT to input a variable for the <PATH>/<FILENAME> of the kubelet config file.
Please set $kubelet_config=<PATH> based on the file location on your system
for example:

```
export kubelet_config=/etc/kubernetes/kubelet.conf
```

To perform the audit manually:
Run the below command (based on the file location on your system) on the each worker node. For example,

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

Verify that the ownership is set to `root:root`.Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /etc/kubernetes/kubelet.conf
```

Default Value:

By default, `kubelet.conf` file has permissions of `600`.

References:

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

CIS Controls:

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

4.1.6 Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1 - Worker Node

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:

Automated AAC auditing has been modified to allow CIS-CAT to input a variable for the <PATH>/<FILENAME> of the kubelet config file.
Please set $kubelet_config=<PATH> based on the file location on your system
for example:

```
export kubelet_config=/etc/kubernetes/kubelet.conf
```

To perform the audit manually:
Run the below command (based on the file location on your system) on the each worker node. For example,

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

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

Remediation:

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

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

Default Value:

By default, `kubelet.conf` file ownership is set to `root:root`.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

4.1.7 Ensure that the certificate authorities file permissions are set to 600 or more restrictive (Manual)

Profile Applicability:

- Level 1 - Worker Node

Description:

Ensure that the certificate authorities file has permissions of `600` 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:

Run the following command:

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

Find the file specified by the `--client-ca-file` argument.
Run the following command:

```
stat -c %a <filename>
```

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

Remediation:

Run the following command to modify the file permissions of the `--client-ca-file`

```
chmod 600 <filename>
```

Default Value:

By default no `--client-ca-file` is specified.

References:

1. https://kubernetes.io/docs/admin/authentication/#x509-client-certs

CIS Controls:

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

4.1.8 Ensure that the client certificate authorities file ownership is set to root:root (Manual)

Profile Applicability:

- Level 1 - Worker Node

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:

Run the following command:

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

Find the file specified by the `--client-ca-file` argument.
Run the following command:

```
stat -c %U:%G <filename>
```

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

Remediation:

Run the following command to modify the ownership of the `--client-ca-file`.

```
chown root:root <filename>
```

Default Value:

By default no `--client-ca-file` is specified.

References:

1. https://kubernetes.io/docs/admin/authentication/#x509-client-certs

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

4.1.9 If the kubelet config.yaml configuration file is being used validate permissions set to 600 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Worker Node

Description:

Ensure that if the kubelet refers to a configuration file with the `--config` argument, that file has permissions of 600 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:

Automated AAC auditing has been modified to allow CIS-CAT to input a variable for the <PATH>/<FILENAME> of the kubelet config yaml file.
Please set $kubelet_config_yaml=<PATH> based on the file location on your system for example:

```
export kubelet_config_yaml=/var/lib/kubelet/config.yaml
```

To perform the audit manually:
Run the below command (based on the file location on your system) on the each worker node. For example,

```
stat -c %a /var/lib/kubelet/config.yaml
```

Verify that the permissions are `600` or more restrictive.

Remediation:

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

```
chmod 600 /var/lib/kubelet/config.yaml
```

Default Value:

By default, the /var/lib/kubelet/config.yaml file as set up by `kubeadm` has permissions of 600.

References:

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

CIS Controls:

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

4.1.10 If the kubelet config.yaml configuration file is being used validate file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1 - Worker Node

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:

Automated AAC auditing has been modified to allow CIS-CAT to input a variable for the <PATH>/<FILENAME> of the kubelet config yaml file.
Please set $kubelet_config_yaml=<PATH> based on the file location on your system for example:

```
export kubelet_config_yaml=/var/lib/kubelet/config.yaml
```

To perform the audit manually:
Run the below command (based on the file location on your system) on the each worker node. For example,

```
stat -c %aU %G /var/lib/kubelet/config.yaml
```
```Verify that the ownership is set to `root:root`.
```

Remediation:

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

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

Default Value:

By default, `/var/lib/kubelet/config.yaml` file as set up by `kubeadm` is owned by `root:root`.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

4.2 Kubelet

This section contains recommendations for kubelet configuration.

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

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

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

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

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

Profile Applicability:

- Level 1 - Worker Node

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:

If using a Kubelet configuration file, check that there is an entry for `authentication: anonymous: enabled` set to `false`.
Run the following command on each node:

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

Verify that the `--anonymous-auth` argument is set to `false`.
This executable argument may be omitted, provided there is a corresponding entry set to `false` in the Kubelet config file.

Remediation:

If using a Kubelet config file, edit the file to set `authentication: anonymous: enabled` to `false`.
If using executable arguments, edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

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

Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, anonymous access is enabled.

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Worker Node

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:

Run the following command on each node:

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

If the `--authorization-mode` argument is present check that it is not set to `AlwaysAllow`.
If it is not present check that there is a Kubelet config file specified by `--config`, and that file sets `authorization: mode` to something other than `AlwaysAllow`.
It is also possible to review the running configuration of a Kubelet via the `/configz` endpoint on the Kubelet API port (typically `10250/TCP`). Accessing these with appropriate credentials will provide details of the Kubelet's configuration.

Remediation:

If using a Kubelet config file, edit the file to set `authorization: mode` to `Webhook`.
If using executable arguments, edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and set the below parameter in `KUBELET_AUTHZ_ARGS` variable.

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

Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, `--authorization-mode` argument is set to `AlwaysAllow`.

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Worker Node

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:

Run the following command on each node:

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

Verify that the `--client-ca-file` argument exists and is set to the location of the client certificate authority file.
If the `--client-ca-file` argument is not present, check that there is a Kubelet config file specified by `--config`, and that the file sets `authentication: x509: clientCAFile` to the location of the client certificate authority file.

Remediation:

If using a Kubelet config file, edit the file to set `authentication: x509: clientCAFile` to the location of the client CA file.
If using command line arguments, edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and set the below parameter in `KUBELET_AUTHZ_ARGS` variable.

```
--client-ca-file=<path/to/client-ca-file>
```

Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, `--client-ca-file` argument is not set.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/

CIS Controls:

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

4.2.4 Verify that the --read-only-port argument is set to 0 (Manual)

Profile Applicability:

- Level 1 - Worker Node

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:

Run the following command on each node:

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

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

Remediation:

If using a Kubelet config file, edit the file to set `readOnlyPort` to `0`.
If using command line arguments, edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

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

Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, `--read-only-port` is set to `10255/TCP`. However, if a config file is specified by `--config` the default value for `readOnlyPort` is 0.

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Worker Node

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:

Run the following command on each node:

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

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:

If using a Kubelet config file, edit the file to set `streamingConnectionIdleTimeout` to a value other than 0.
If using command line arguments, edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

```
--streaming-connection-idle-timeout=5m
```

Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, `--streaming-connection-idle-timeout` is set to 4 hours.

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Worker Node

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:

Run the following command on each node:

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

Verify that if the `--make-iptables-util-chains` argument exists then it is set to `true`.
If the `--make-iptables-util-chains` argument does not exist, and there is a Kubelet config file specified by `--config`, verify that the file does not set `makeIPTablesUtilChains` to `false`.

Remediation:

If using a Kubelet config file, edit the file to set `makeIPTablesUtilChains: true`.
If using command line arguments, edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and remove the `--make-iptables-util-chains` argument from the `KUBELET_SYSTEM_PODS_ARGS` variable.
Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, `--make-iptables-util-chains` argument is set to `true`.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	2.5 Allowlist Authorized Software Use technical controls, such as application allowlisting, to ensure that only authorized software can execute or be accessed. Reassess bi-annually, or more frequently.		●	●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

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

Profile Applicability:

- Level 1 - Worker Node

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:

Run the following command on each node:

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

Verify that `--hostname-override` argument does not exist.
Note This setting is not configurable via the Kubelet config file.

Remediation:

Edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and remove the `--hostname-override` argument from the `KUBELET_SYSTEM_PODS_ARGS` variable.
Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, `--hostname-override` argument is not set.

References:

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

2. https://github.com/kubernetes/kubernetes/issues/22063

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4 Secure Configuration of Enterprise Assets and Software Establish and maintain the secure configuration of enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/IoT devices; and servers) and software (operating systems and applications).			
v7	5 Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers			

4.2.8 Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture (Manual)

Profile Applicability:

- Level 2 - Worker Node

Description:

Security relevant information should be captured. The eventRecordQPS on the Kubelet configuration can be used to limit the rate at which events are gathered and sets the maximum event creations per second. 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:

Run the following command on each node:

```
sudo grep "eventRecordQPS" /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Review the value set for the argument and determine whether this has been set to an appropriate level for the cluster.
If the argument does not exist, check that there is a Kubelet config file specified by `--config` and review the value in this location.

Remediation:

If using a Kubelet config file, edit the file to set `eventRecordQPS:` to an appropriate level.
If using command line arguments, edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.
Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, eventRecordQPS argument is set to `5`.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/apis/kubeletconfig/v1beta1/types.go

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Worker Node

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.

Audit:

Run the following command on each node:

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

Verify that the --tls-cert-file and --tls-private-key-file arguments exist and they are set as appropriate.
If these arguments are not present, check that there is a Kubelet config specified by --config and that it contains appropriate settings for tlsCertFile and tlsPrivateKeyFile.

Remediation:

If using a Kubelet config file, edit the file to set tlsCertFile to the location of the certificate file to use to identify this Kubelet, and tlsPrivateKeyFile to the location of the corresponding private key file.
If using command line arguments, edit the kubelet service file /etc/kubernetes/kubelet.conf on each worker node and set the below parameters in KUBELET_CERTIFICATE_ARGS variable.
--tls-cert-file=<path/to/tls-certificate-file> --tls-private-key-file=<path/to/tls-key-file>
Based on your system, restart the kubelet service. For example:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Worker Node

Description:

Enable kubelet client certificate rotation.

Rationale:

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

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

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

Impact:

None

Audit:

Run the following command on each node:

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

Verify that the `--rotate-certificates` argument is not present, or is set to `true`.
If the `--rotate-certificates` argument is not present, verify that if there is a Kubelet config file specified by `--config`, that file does not contain `rotateCertificates: false`.

Remediation:

If using a Kubelet config file, edit the file to add the line `rotateCertificates: true` or remove it altogether to use the default value.
If using command line arguments, edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and remove `--rotate-certificates=false` argument from the `KUBELET_CERTIFICATE_ARGS` variable.
Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, kubelet client certificate rotation is enabled.

References:

1. https://github.com/kubernetes/kubernetes/pull/41912
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-tls-bootstrapping/#kubelet-configuration
3. https://kubernetes.io/docs/imported/release/notes/
4. https://kubernetes.io/docs/reference/command-line-tools-reference/feature-gates/

CIS Controls:

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

4.2.11 Verify that the RotateKubeletServerCertificate argument is set to true (Manual)

Profile Applicability:

- Level 1 - Worker Node

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:

Ignore this check if serverTLSBootstrap is true in the kubelet config file or if the --rotate-server-certificates parameter is set on kubelet
Run the following command on each node:

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

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

Remediation:

Edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and set the below parameter in `KUBELET_CERTIFICATE_ARGS` variable.

```
--feature-gates=RotateKubeletServerCertificate=true
```

Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default, kubelet server certificate rotation is enabled.

References:

1. https://github.com/kubernetes/kubernetes/pull/45059
2. https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/#kubelet-configuration

CIS Controls:

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

4.2.12 Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)

Profile Applicability:

- Level 1 - Worker Node

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`

Run the following command on each node:

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

If the `--tls-cipher-suites` argument is present, ensure it only contains values included in this set.
If it is not present check that there is a Kubelet config file specified by `--config`, and that file sets `TLSCipherSuites:` to only include values from this set.

Remediation:

If using a Kubelet config file, edit the file to set `TLSCipherSuites:` to `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` or to a subset of these values.
If using executable arguments, edit the kubelet service file `/etc/kubernetes/kubelet.conf` on each worker node and set the `--tls-cipher-suites` parameter as follows, or to a subset of these values.

```
 --tls-cipher-
suites=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
```

Based on your system, restart the `kubelet` service. For example:

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

Default Value:

By default the Kubernetes API server supports a wide range of TLS ciphers

Additional Information:

The list chosen above should be fine for modern clients. It's essentially the list from the Mozilla "Modern cipher" option with the ciphersuites supporting CBC mode removed, as CBC has traditionally had a lot of issues

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

4.2.13 Ensure that a limit is set on pod PIDs (Manual)

Profile Applicability:

- Level 1 - Worker Node

Description:

Ensure that the Kubelet sets limits on the number of PIDs that can be created by pods running on the node.

Rationale:

By default pods running in a cluster can consume any number of PIDs, potentially exhausting the resources available on the node. Setting an appropriate limit reduces the risk of a denial of service attack on cluster nodes.

Impact:

Setting this value will restrict the number of processes per pod. If this limit is lower than the number of PIDs required by a pod it will not operate.

Audit:

Review the Kubelet's start-up parameters for the value of `--pod-max-pids`, and check the Kubelet configuration file for the `PodPidsLimit` . If neither of these values is set, then there is no limit in place.

Remediation:

Decide on an appropriate level for this parameter and set it, either via the `--pod-max-pids` command line parameter or the `PodPidsLimit` configuration file setting.

Default Value:

By default the number of PIDs is not limited.

References:

1. https://kubernetes.io/docs/concepts/policy/pid-limiting/#pod-pid-limits

4.3 kube-proxy

Recommendations relating to the kube-proxy component.

4.3.1 Ensure that the kube-proxy metrics service is bound to localhost (Automated)

Profile Applicability:

- Level 1 - Worker Node

Description:

Do not bind the kube-proxy metrics port to non-loopback addresses.

Rationale:

kube-proxy has two APIs which provided access to information about the service and can be bound to network ports. The metrics API service includes endpoints (`/metrics` and `/configz`) which disclose information about the configuration and operation of kube-proxy. These endpoints should not be exposed to untrusted networks as they do not support encryption or authentication to restrict access to the data they provide.

Impact:

3rd party services which try to access metrics or configuration information related to kube-proxy will require access to the localhost interface of the node.

Audit:

review the start-up flags provided to kube proxy

```
ps -ef | grep -i kube-proxy
```

Ensure that the `--metrics-bind-address` parameter is not set to a value other than 127.0.0.1. From the output of this command gather the location specified in the `--config` parameter. Review any file stored at that location and ensure that it does not specify a value other than 127.0.0.1 for `metricsBindAddress`.

Remediation:

Modify or remove any values which bind the metrics service to a non-localhost address

Default Value:

The default value is `127.0.0.1:10249`

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.6 Use of Secure Network Management and Communication Protocols Use secure network management and communication protocols (e.g., 802.1X, Wi-Fi Protected Access 2 (WPA2) Enterprise or greater).		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●

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 (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

Impact:

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

Audit:

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

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

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

Remediation:

Identify all clusterrolebindings to the cluster-admin role. Check if they are used and if they need this role or if they could use a role with fewer privileges.
Where possible, first bind users to a lower privileged role and then remove the clusterrolebinding to the cluster-admin role :

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

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/authorization/rbac/#user-facing-roles

CIS Controls:

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

5.1.2 Minimize access to secrets (Automated)

Profile Applicability:

- Level 1 - Master Node

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 kubeadm cluster the following list of principals have `get` privileges on `secret` objects

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3 Data Protection Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data.			
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

5.1.3 Minimize wildcard use in Roles and ClusterRoles (Automated)

Profile Applicability:

- Level 1 - Worker Node

Description:

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

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

Rationale:

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

Audit:

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

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

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

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

Remediation:

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

CIS Controls:

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

Controls Version	Control	IG 1	IG 2	IG 3
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

5.1.4 Minimize access to create pods (Automated)

Profile Applicability:

- Level 1 - Master Node

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

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

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

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

Impact:

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

Audit:

For each namespace in the cluster, review the rights assigned to the default service account and ensure that it has no roles or cluster roles bound to it apart from the defaults.
Additionally ensure that the `automountServiceAccountToken: false` setting is in place for each default service account.

Remediation:

Create explicit service accounts wherever a Kubernetes workload requires specific access to the Kubernetes API server.
Modify the configuration of each default service account to include this value

```
automountServiceAccountToken: false
```

Default Value:

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

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1 - Master Node

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:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3 Data Protection Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data.			
v7	13 Data Protection Data Protection			

5.1.7 Avoid use of system:masters group (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

The special group `system:masters` should not be used to grant permissions to any user or service account, except where strictly necessary (e.g. bootstrapping access prior to RBAC being fully available)

Rationale:

The `system:masters` group has unrestricted access to the Kubernetes API hard-coded into the API server source code. An authenticated user who is a member of this group cannot have their access reduced, even if all bindings and cluster role bindings which mention it, are removed.

When combined with client certificate authentication, use of this group can allow for irrevocable cluster-admin level credentials to exist for a cluster.

Impact:

Once the RBAC system is operational in a cluster `system:masters` should not be specifically required, as ordinary bindings from principals to the `cluster-admin` cluster role can be made where unrestricted access is required.

Audit:

Review a list of all credentials which have access to the cluster and ensure that the group `system:masters` is not used.

Remediation:

Remove the `system:masters` group from all users in the cluster.

Default Value:

By default some clusters will create a "break glass" client certificate which is a member of this group. Access to this client certificate should be carefully controlled and it should not be used for general cluster operations.

References:

1. https://github.com/kubernetes/kubernetes/blob/master/pkg/registry/rbac/escalation_check.go#L38

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			
v6	5.1 Minimize And Sparingly Use Administrative Privileges Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.			

5.1.8 Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Cluster roles and roles with the impersonate, bind or escalate permissions should not be granted unless strictly required. Each of these permissions allow a particular subject to escalate their privileges beyond those explicitly granted by cluster administrators

Rationale:

The impersonate privilege allows a subject to impersonate other users gaining their rights to the cluster. The bind privilege allows the subject to add a binding to a cluster role or role which escalates their effective permissions in the cluster. The escalate privilege allows a subject to modify cluster roles to which they are bound, increasing their rights to that level.

Each of these permissions has the potential to allow for privilege escalation to cluster-admin level.

Impact:

There are some cases where these permissions are required for cluster service operation, and care should be taken before removing these permissions from system service accounts.

Audit:

Review the users who have access to cluster roles or roles which provide the impersonate, bind or escalate privileges.

Remediation:

Where possible, remove the impersonate, bind and escalate rights from subjects.

Default Value:

In a default kubeadm cluster, the system:masters group and clusterrole-aggregation-controller service account have access to the escalate privilege. The system:masters group also has access to bind and impersonate.

References:

1. https://www.impidio.com/blog/kubernetes-rbac-security-pitfalls
2. https://raesene.github.io/blog/2020/12/12/Escalating_Away/
3. https://raesene.github.io/blog/2021/01/16/Getting-Into-A-Bind-with-Kubernetes/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

5.1.9 Minimize access to create persistent volumes (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

The ability to create persistent volumes in a cluster can provide an opportunity for privilege escalation, via the creation of `hostPath` volumes. As persistent volumes are not covered by Pod Security Admission, a user with access to create persistent volumes may be able to get access to sensitive files from the underlying host even where restrictive Pod Security Admission policies are in place.

Rationale:

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

Audit:

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

Remediation:

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

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#persistent-volume-creation

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

5.1.10 Minimize access to the proxy sub-resource of nodes (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Users with access to the `Proxy` sub-resource of `Node` objects automatically have permissions to use the Kubelet API, which may allow for privilege escalation or bypass cluster security controls such as audit logs.

The Kubelet provides an API which includes rights to execute commands in any container running on the node. Access to this API is covered by permissions to the main Kubernetes API via the `node` object. The proxy sub-resource specifically allows wide ranging access to the Kubelet API.

Direct access to the Kubelet API bypasses controls like audit logging (there is no audit log of Kubelet API access) and admission control.

Rationale:

The ability to use the `proxy` sub-resource of `node` objects opens up possibilities for privilege escalation and should be restricted, where possible.

Audit:

Review the users who have access to the `proxy` sub-resource of `node` objects in the Kubernetes API.

Remediation:

Where possible, remove access to the `proxy` sub-resource of `node` objects.

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#access-to-proxy-subresource-of-nodes
2. https://kubernetes.io/docs/reference/access-authn-authz/kubelet-authn-authz/#kubelet-authorization

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

5.1.11 Minimize access to the approval sub-resource of certificatesigningrequests objects (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Users with access to the update the `approval` sub-resource of `certificateaigningrequest` objects can approve new client certificates for the Kubernetes API effectively allowing them to create new high-privileged user accounts.

This can allow for privilege escalation to full cluster administrator, depending on users configured in the cluster

Rationale:

The ability to update certificate signing requests should be limited.

Audit:

Review the users who have access to update the `approval` sub-resource of `certificatesigningrequest` objects in the Kubernetes API.

Remediation:

Where possible, remove access to the `approval` sub-resource of `certificatesigningrequest` objects.

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#csrs-and-certificate-issuing

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

5.1.12 Minimize access to webhook configuration objects (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Users with rights to create/modify/delete `validatingwebhookconfigurations` or `mutatingwebhookconfigurations` can control webhooks that can read any object admitted to the cluster, and in the case of mutating webhooks, also mutate admitted objects. This could allow for privilege escalation or disruption of the operation of the cluster.

Rationale:

The ability to manage webhook configuration should be limited

Audit:

Review the users who have access to `validatingwebhookconfigurations` or `mutatingwebhookconfigurations` objects in the Kubernetes API.

Remediation:

Where possible, remove access to the `validatingwebhookconfigurations` or `mutatingwebhookconfigurations` objects

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#control-admission-webhooks

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

5.1.13 Minimize access to the service account token creation (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Users with rights to create new service account tokens at a cluster level, can create long-lived privileged credentials in the cluster. This could allow for privilege escalation and persistent access to the cluster, even if the users account has been revoked.

Rationale:

The ability to create service account tokens should be limited.

Audit:

Review the users who have access to create the `token` sub-resource of `serviceaccount` objects in the Kubernetes API.

Remediation:

Where possible, remove access to the `token` sub-resource of `serviceaccount` objects.

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#token-request

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

5.2 Pod Security Standards

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

5.2.1 Ensure that the cluster has at least one active policy control mechanism in place (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Every Kubernetes cluster should have at least one policy control mechanism in place to enforce the other requirements in this section. This could be the in-built Pod Security Admission controller, or a third party policy control system.

Rationale:

Without an active policy control mechanism, it is not possible to limit the use of containers with access to underlying cluster nodes, via mechanisms like privileged containers, or the use of hostPath volume mounts.

Impact:

Where policy control systems are in place, there is a risk that workloads required for the operation of the cluster may be stopped from running. Care is required when implementing admission control policies to ensure that this does not occur.

Audit:

Review the workloads deployed to the cluster to understand if Pod Security Admission or external admission control systems are in place.

Remediation:

Ensure that either Pod Security Admission or an external policy control system is in place for every namespace which contains user workloads.

Default Value:

By default, Pod Security Admission is enabled but no policies are in place.

References:

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

5.2.2 Minimize the admission of privileged containers (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

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

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

Impact:

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

Audit:

Run the following command:

```
get pods -A -o=jsonpath=$'{range .items[*]}{@.metadata.name}:
{@..securityContext}\n{end}'
```

It will produce an inventory of all the privileged use on the cluster, if any (please, refer to a sample below). Further grepping can be done to automate each specific violation detection.
calico-kube-controllers-57b57c56f-jtmk4: {} << No Elevated Privileges calico-node-c4xv4: {} {"privileged":true} {"privileged":true} {"privileged":true} {"privileged":true} << Violates 5.2.2 dashboard-metrics-scraper-7bc864c59-2m2xw: {"seccompProfile":{"type":"RuntimeDefault"}} {"allowPrivilegeEscalation":false,"readOnlyRootFilesystem":true,"runAsGroup":2001,"runAsUser":1001}

Remediation:

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

Default Value:

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

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			
v6	5.1 Minimize And Sparingly Use Administrative Privileges Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.			

5.2.3 Minimize the admission of containers wishing to share the host process ID namespace (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

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

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

Impact:

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

Audit:

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

Remediation:

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

Default Value:

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

References:

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

CIS Controls:

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

5.2.4 Minimize the admission of containers wishing to share the host IPC namespace (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

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

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

Impact:

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

Audit:

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

Remediation:

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

Default Value:

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

References:

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

CIS Controls:

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

5.2.5 Minimize the admission of containers wishing to share the host network namespace (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

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

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

Impact:

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

Audit:

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

Remediation:

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

Default Value:

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

References:

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

CIS Controls:

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

5.2.6 Minimize the admission of containers with allowPrivilegeEscalation (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

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

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

Rationale:

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

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

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

Impact:

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

Audit:

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

Remediation:

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

Default Value:

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

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

5.2.7 Minimize the admission of root containers (Manual)

Profile Applicability:

- Level 2 - Master Node

Description:

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

Rationale:

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

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

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

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

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy restricts the use of root containers by setting `MustRunAsNonRoot` or `MustRunAs` with the range of UIDs not including 0.

Remediation:

Create a policy for each namespace in the cluster, ensuring that either `MustRunAsNonRoot` or `MustRunAs` with the range of UIDs not including 0, is set.

Default Value:

By default, there are no restrictions on the use of root containers and if a User is not specified in the image, the container will run as root.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

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

Profile Applicability:

- Level 1 - Master Node

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 admission control policy defined which does not permit containers with the NET_RAW capability.

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

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that at least one policy disallows the admission of containers with the `NET_RAW` capability.

Remediation:

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

Default Value:

By default, there are no restrictions on the creation of containers with the `NET_RAW` capability.

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-standards/
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

5.2.9 Minimize the admission of containers with added capabilities (Manual)

Profile Applicability:

- Level 1 - Master Node

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 policy 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 policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that policies are present which prevent `allowedCapabilities` to be set to anything other than an empty array.

Remediation:

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

Default Value:

By default, there are no restrictions on adding capabilities to containers.

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-standards/
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

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

Profile Applicability:

- Level 2 - Master Node

Description:

Do not generally permit containers with capabilities

Rationale:

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

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

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that at least one policy requires that capabilities are dropped by all containers.

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 policy which forbids the admission of containers which do not drop all capabilities.

Default Value:

By default, there are no restrictions on the creation of containers with additional capabilities

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-standards/
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

5.2.11 Minimize the admission of Windows HostProcess Containers (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

A Windows container making use of the `hostProcess` flag can interact with the underlying Windows cluster node. As per the Kubernetes documentation, this provides "privileged access" to the Windows node.

Where Windows containers are used inside a Kubernetes cluster, there should be at least one admission control policy which does not permit `hostProcess` Windows containers.

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

Impact:

Pods defined with `securityContext.windowsOptions.hostProcess: true` will not be permitted unless they are run under a specific policy.

Audit:

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

Remediation:

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

Default Value:

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

References:

1. https://kubernetes.io/docs/tasks/configure-pod-container/create-hostprocess-pod/
2. https://kubernetes.io/docs/concepts/security/pod-security-standards/

5.2.12 Minimize the admission of HostPath volumes (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Do not generally admit containers which make use of `hostPath` volumes.

Rationale:

A container which mounts a `hostPath` volume as part of its specification will have access to the filesystem of the underlying cluster node. The use of `hostPath` volumes may allow containers access to privileged areas of the node filesystem.

There should be at least one admission control policy defined which does not permit containers to mount `hostPath` volumes.

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

Impact:

Pods defined which make use of `hostPath` volumes will not be permitted unless they are run under a spefific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of containers with `hostPath` volumes.

Remediation:

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

Default Value:

By default, there are no restrictions on the creation of `hostPath` volumes.

References:

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

5.2.13 Minimize the admission of containers which use HostPorts (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

Do not generally permit containers which require the use of HostPorts.

Rationale:

Host ports connect containers directly to the host's network. This can bypass controls such as network policy.

There should be at least one admission control policy defined which does not permit containers which require the use of HostPorts.

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

Impact:

Pods defined with `hostPort` settings in either the container, initContainer or ephemeralContainer sections will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of containers which have `hostPort` sections.

Remediation:

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

Default Value:

By default, there are no restrictions on the use of HostPorts.

References:

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

5.3 Network Policies and CNI

5.3.1 Ensure that the CNI in use supports Network Policies (Manual)

Profile Applicability:

- Level 1 - Master Node

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.

Remediation:

If the CNI plugin in use does not support network policies, consideration should be given to making use of a different plugin, or finding an alternate mechanism for restricting traffic in the Kubernetes cluster.

Default Value:

This will depend on the CNI plugin in use.

References:

1. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/

Additional Information:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	9.5 Implement Application Firewalls Place application firewalls in front of any critical servers to verify and validate the traffic going to the server. Any unauthorized traffic should be blocked and logged.			●

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

Profile Applicability:

- Level 2 - Master Node

Description:

Use network policies to isolate traffic in your cluster network.

Rationale:

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

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

Impact:

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

Audit:

Run the below command and review the `NetworkPolicy` objects created in the cluster.

```
kubectl get networkpolicy --all-namespaces
```

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

Remediation:

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

Default Value:

By default, network policies are not created.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	14.2 Enable Firewall Filtering Between VLANs Enable firewall filtering between VLANs to ensure that only authorized systems are able to communicate with other systems necessary to fulfill their specific responsibilities.		●	●

5.4 Secrets Management

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

Profile Applicability:

- Level 2 - Master Node

Description:

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

Rationale:

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

Impact:

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

Audit:

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

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

Remediation:

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

Default Value:

By default, secrets are not defined

References:

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

Additional Information:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3 Data Protection Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data.			
v7	13 Data Protection Data Protection			

5.4.2 Consider external secret storage (Manual)

Profile Applicability:

- Level 2 - Master Node

Description:

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

Rationale:

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

Impact:

None

Audit:

Review your secrets management implementation.

Remediation:

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

Default Value:

By default, no external secret management is configured.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3 Data Protection Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data.			
v7	13 Data Protection Data Protection			

5.5 Extensible Admission Control

5.5.1 Configure Image Provenance using ImagePolicyWebhook admission controller (Manual)

Profile Applicability:

- Level 2 - Master Node

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.

Impact:

You need to regularly maintain your provenance configuration based on container image updates.

Audit:

Review the pod definitions in your cluster and verify that image provenance is configured as appropriate.

Remediation:

Follow the Kubernetes documentation and setup image provenance.

Default Value:

By default, image provenance is not set.

References:

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

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.7 Use Standard Hardening Configuration Templates for Application Infrastructure Use standard, industry-recommended hardening configuration templates for application infrastructure components. This includes underlying servers, databases, and web servers, and applies to cloud containers, Platform as a Service (PaaS) components, and SaaS components. Do not allow in-house developed software to weaken configuration hardening.		●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

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 - Master Node

Description:

Use namespaces to isolate your Kubernetes objects.

Rationale:

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

Impact:

You need to switch between namespaces for administration.

Audit:

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

```
kubectl get namespaces
```

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

Remediation:

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

Default Value:

By default, Kubernetes starts with 4 initial namespaces:

1. `default` - The default namespace for objects with no other namespace
2. `kube-system` - The namespace for objects created by the Kubernetes system
3. `kube-node-lease` - Namespace used for node heartbeats
4. `kube-public` - Namespace used for public information in a cluster

References:

1. https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/#viewing-namespaces
2. http://blog.kubernetes.io/2016/08/security-best-practices-kubernetes-deployment.html
3. https://github.com/kubernetes/enhancements/tree/master/keps/sig-node/589-efficient-node-heartbeats

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13 Network Monitoring and Defense Operate processes and tooling to establish and maintain comprehensive network monitoring and defense against security threats across the enterprise's network infrastructure and user base.			
v7	12 Boundary Defense Boundary Defense			

5.7.2 Ensure that the seccomp profile is set to docker/default in your pod definitions (Manual)

Profile Applicability:

- Level 2 - Master Node

Description:

Enable `docker/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 `docker/default` seccomp profile is too restrictive for you, you would have to create/manage your own seccomp profiles.

Audit:

Review the pod definitions in your cluster. It should create a line as below:

```
  securityContext:
    seccompProfile:
      type: RuntimeDefault
```

Remediation:

Use security context to enable the `docker/default` seccomp profile in your pod definitions. An example is as below:

```
  securityContext:
    seccompProfile:
      type: RuntimeDefault
```

Default Value:

By default, seccomp profile is set to `unconfined` which means that no seccomp profiles are enabled.

References:

1. https://kubernetes.io/docs/tutorials/clusters/seccomp/
2. https://docs.docker.com/engine/security/seccomp/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.7 Use Standard Hardening Configuration Templates for Application Infrastructure Use standard, industry-recommended hardening configuration templates for application infrastructure components. This includes underlying servers, databases, and web servers, and applies to cloud containers, Platform as a Service (PaaS) components, and SaaS components. Do not allow in-house developed software to weaken configuration hardening.		●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

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

Profile Applicability:

- Level 2 - Master Node

Description:

Apply Security Context to Your Pods and Containers

Rationale:

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

Impact:

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

Audit:

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

Remediation:

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:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4 Secure Configuration of Enterprise Assets and Software Establish and maintain the secure configuration of enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/IoT devices; and servers) and software (operating systems and applications).			
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

5.7.4 The default namespace should not be used (Manual)

Profile Applicability:

- Level 2 - Master Node

Description:

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

Rationale:

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

Impact:

None

Audit:

Run this command to list objects in default namespace

```
kubectl get $(kubectl api-resources --verbs=list --namespaced=true -o name | paste -sd, -) --ignore-not-found -n default
```

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

Remediation:

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

Default Value:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.2 Establish and Maintain a Secure Network Architecture Establish and maintain a secure network architecture. A secure network architecture must address segmentation, least privilege, and availability, at a minimum.		●	●

Controls Version	Control	IG 1	IG 2	IG 3
v7	2.10 Physically or Logically Segregate High Risk Applications Physically or logically segregated systems should be used to isolate and run software that is required for business operations but incur higher risk for the organization.			●

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**Control Plane Components**		
1.1	**Control Plane Node Configuration Files**		
1.1.1	Ensure that the API server pod specification file permissions are set to 600 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 600 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 600 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 600 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 600 or more restrictive (Manual)	☐	☐
1.1.10	Ensure that the Container Network Interface file ownership is set to root:root (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
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 default administrative credential file permissions are set to 600 (Automated)	☐	☐
1.1.14	Ensure that the default administrative credential file ownership is set to root:root (Automated)	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 600 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 600 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 Kubernetes PKI directory and file ownership is set to root:root (Automated)	☐	☐
1.1.20	Ensure that the Kubernetes PKI certificate file permissions are set to 600 or more restrictive (Manual)	☐	☐
1.1.21	Ensure that the Kubernetes PKI key file permissions are set to 600 (Manual)	☐	☐
1.2	**API Server**		
1.2.1	Ensure that the --anonymous-auth argument is set to false (Manual)	☐	☐
1.2.2	Ensure that the --token-auth-file parameter is not set (Automated)	☐	☐
1.2.3	Ensure that the DenyServiceExternalIPs is set (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1.2.4	Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Automated)	☐	☐
1.2.5	Ensure that the --kubelet-certificate-authority argument is set as appropriate (Automated)	☐	☐
1.2.6	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	☐	☐
1.2.7	Ensure that the --authorization-mode argument includes Node (Automated)	☐	☐
1.2.8	Ensure that the --authorization-mode argument includes RBAC (Automated)	☐	☐
1.2.9	Ensure that the admission control plugin EventRateLimit is set (Manual)	☐	☐
1.2.10	Ensure that the admission control plugin AlwaysAdmit is not set (Automated)	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysPullImages is set (Manual)	☐	☐
1.2.12	Ensure that the admission control plugin ServiceAccount is set (Automated)	☐	☐
1.2.13	Ensure that the admission control plugin NamespaceLifecycle is set (Automated)	☐	☐
1.2.14	Ensure that the admission control plugin NodeRestriction is set (Automated)	☐	☐
1.2.15	Ensure that the --profiling argument is set to false (Automated)	☐	☐
1.2.16	Ensure that the --audit-log-path argument is set (Automated)	☐	☐
1.2.17	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1.2.18	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Automated)	☐	☐
1.2.19	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Automated)	☐	☐
1.2.20	Ensure that the --request-timeout argument is set as appropriate (Manual)	☐	☐
1.2.21	Ensure that the --service-account-lookup argument is set to true (Automated)	☐	☐
1.2.22	Ensure that the --service-account-key-file argument is set as appropriate (Automated)	☐	☐
1.2.23	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Automated)	☐	☐
1.2.24	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	☐	☐
1.2.25	Ensure that the --client-ca-file argument is set as appropriate (Automated)	☐	☐
1.2.26	Ensure that the --etcd-cafile argument is set as appropriate (Automated)	☐	☐
1.2.27	Ensure that the --encryption-provider-config argument is set as appropriate (Manual)	☐	☐
1.2.28	Ensure that encryption providers are appropriately configured (Manual)	☐	☐
1.2.29	Ensure that the API Server only makes use of Strong Cryptographic Ciphers (Manual)	☐	☐
1.3	**Controller Manager**		
1.3.1	Ensure that the --terminated-pod-gc-threshold argument is set as appropriate (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1.3.2	Ensure that the --profiling argument is set to false (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 --profiling argument is set to false (Automated)	☐	☐
1.4.2	Ensure that the --bind-address argument is set to 127.0.0.1 (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)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
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)	☐	☐
3.1.2	Service account token authentication should not be used for users (Manual)	☐	☐
3.1.3	Bootstrap token authentication should not be used for users (Manual)	☐	☐
3.2	**Logging**		
3.2.1	Ensure that a minimal audit policy is created (Manual)	☐	☐
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 600 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 600 or more restrictive (Manual)	☐	☐
4.1.4	If proxy kubeconfig file exists ensure ownership is set to root:root (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 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 600 or more restrictive (Manual)	☐	☐
4.1.8	Ensure that the client certificate authorities file ownership is set to root:root (Manual)	☐	☐
4.1.9	If the kubelet config.yaml configuration file is being used validate permissions set to 600 or more restrictive (Automated)	☐	☐
4.1.10	If the kubelet config.yaml configuration file is being used validate 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 argument is set to 0 (Manual)	☐	☐
4.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Manual)	☐	☐
4.2.6	Ensure that the --make-iptables-util-chains argument is set to true (Automated)	☐	☐
4.2.7	Ensure that the --hostname-override argument is not set (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.2.8	Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture (Manual)	☐	☐
4.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Manual)	☐	☐
4.2.10	Ensure that the --rotate-certificates argument is not set to false (Automated)	☐	☐
4.2.11	Verify that the RotateKubeletServerCertificate argument is set to true (Manual)	☐	☐
4.2.12	Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)	☐	☐
4.2.13	Ensure that a limit is set on pod PIDs (Manual)	☐	☐
4.3	**kube-proxy**		
4.3.1	Ensure that the kube-proxy metrics service is bound to localhost (Automated)	☐	☐
5	**Policies**		
5.1	**RBAC and Service Accounts**		
5.1.1	Ensure that the cluster-admin role is only used where required (Automated)	☐	☐
5.1.2	Minimize access to secrets (Automated)	☐	☐
5.1.3	Minimize wildcard use in Roles and ClusterRoles (Automated)	☐	☐
5.1.4	Minimize access to create pods (Automated)	☐	☐
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 (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.1.7	Avoid use of system:masters group (Manual)	☐	☐
5.1.8	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster (Manual)	☐	☐
5.1.9	Minimize access to create persistent volumes (Manual)	☐	☐
5.1.10	Minimize access to the proxy sub-resource of nodes (Manual)	☐	☐
5.1.11	Minimize access to the approval sub-resource of certificatesigningrequests objects (Manual)	☐	☐
5.1.12	Minimize access to webhook configuration objects (Manual)	☐	☐
5.1.13	Minimize access to the service account token creation (Manual)	☐	☐
5.2	**Pod Security Standards**		
5.2.1	Ensure that the cluster has at least one active policy control mechanism in place (Manual)	☐	☐
5.2.2	Minimize the admission of privileged containers (Manual)	☐	☐
5.2.3	Minimize the admission of containers wishing to share the host process ID namespace (Manual)	☐	☐
5.2.4	Minimize the admission of containers wishing to share the host IPC namespace (Manual)	☐	☐
5.2.5	Minimize the admission of containers wishing to share the host network namespace (Manual)	☐	☐
5.2.6	Minimize the admission of containers with allowPrivilegeEscalation (Manual)	☐	☐
5.2.7	Minimize the admission of root containers (Manual)	☐	☐
5.2.8	Minimize the admission of containers with the NET_RAW capability (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.2.9	Minimize the admission of containers with added capabilities (Manual)	☐	☐
5.2.10	Minimize the admission of containers with capabilities assigned (Manual)	☐	☐
5.2.11	Minimize the admission of Windows HostProcess Containers (Manual)	☐	☐
5.2.12	Minimize the admission of HostPath volumes (Manual)	☐	☐
5.2.13	Minimize the admission of containers which use HostPorts (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 (Manual)	☐	☐
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 ImagePolicyWebhook admission controller (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)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.7.3	Apply Security Context to Your Pods and Containers (Manual)	☐	☐
5.7.4	The default namespace should not be used (Manual)	☐	☐

Appendix: CIS Controls v7 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1.1	Ensure that the API server pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.3	Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.5	Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.7	Ensure that the etcd pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.9	Ensure that the Container Network Interface file permissions are set to 600 or more restrictive	☐	☐
1.1.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive	☐	☐
1.1.13	Ensure that the default administrative credential file permissions are set to 600	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.17	Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.19	Ensure that the Kubernetes PKI directory and file ownership is set to root:root	☐	☐
1.1.21	Ensure that the Kubernetes PKI key file permissions are set to 600	☐	☐
1.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
1.2.8	Ensure that the --authorization-mode argument includes RBAC	☐	☐
1.2.10	Ensure that the admission control plugin AlwaysAdmit is not set	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysPullImages is set	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.2.12	Ensure that the admission control plugin ServiceAccount is set	☐	☐
1.2.13	Ensure that the admission control plugin NamespaceLifecycle is set	☐	☐
1.2.16	Ensure that the --audit-log-path argument is set	☐	☐
1.2.20	Ensure that the --request-timeout argument is set as appropriate	☐	☐
1.2.29	Ensure that the API Server only makes use of Strong Cryptographic Ciphers	☐	☐
1.3.1	Ensure that the --terminated-pod-gc-threshold argument is set as appropriate	☐	☐
2.5	Ensure that the --peer-client-cert-auth argument is set to true	☐	☐
3.2.1	Ensure that a minimal audit policy is created	☐	☐
4.1.1	Ensure that the kubelet service file permissions are set to 600 or more restrictive	☐	☐
4.1.3	If proxy kubeconfig file exists ensure permissions are set to 600 or more restrictive	☐	☐
4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 or more restrictive	☐	☐
4.1.7	Ensure that the certificate authorities file permissions are set to 600 or more restrictive	☐	☐
4.1.9	If the kubelet config.yaml configuration file is being used validate permissions set to 600 or more restrictive	☐	☐
4.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
4.2.8	Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture	☐	☐
4.2.12	Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers	☐	☐
5.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
5.1.5	Ensure that default service accounts are not actively used.	☐	☐
5.7.3	Apply Security Context to Your Pods and Containers	☐	☐

Appendix: CIS Controls v7 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1.1	Ensure that the API server pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.3	Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.5	Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.7	Ensure that the etcd pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.9	Ensure that the Container Network Interface file permissions are set to 600 or more restrictive	☐	☐
1.1.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive	☐	☐
1.1.13	Ensure that the default administrative credential file permissions are set to 600	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.17	Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.19	Ensure that the Kubernetes PKI directory and file ownership is set to root:root	☐	☐
1.1.21	Ensure that the Kubernetes PKI key file permissions are set to 600	☐	☐
1.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
1.2.2	Ensure that the --token-auth-file parameter is not set	☐	☐
1.2.3	Ensure that the DenyServiceExternalIPs is set	☐	☐
1.2.6	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
1.2.7	Ensure that the --authorization-mode argument includes Node	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.2.8	Ensure that the --authorization-mode argument includes RBAC	☐	☐
1.2.9	Ensure that the admission control plugin EventRateLimit is set	☐	☐
1.2.10	Ensure that the admission control plugin AlwaysAdmit is not set	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysPullImages is set	☐	☐
1.2.12	Ensure that the admission control plugin ServiceAccount is set	☐	☐
1.2.13	Ensure that the admission control plugin NamespaceLifecycle is set	☐	☐
1.2.16	Ensure that the --audit-log-path argument is set	☐	☐
1.2.17	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate	☐	☐
1.2.18	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate	☐	☐
1.2.19	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate	☐	☐
1.2.20	Ensure that the --request-timeout argument is set as appropriate	☐	☐
1.2.22	Ensure that the --service-account-key-file argument is set as appropriate	☐	☐
1.2.24	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
1.2.25	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
1.2.29	Ensure that the API Server only makes use of Strong Cryptographic Ciphers	☐	☐
1.3.1	Ensure that the --terminated-pod-gc-threshold argument is set as appropriate	☐	☐
1.3.2	Ensure that the --profiling argument is set to false	☐	☐
1.3.3	Ensure that the --use-service-account-credentials argument is set to true	☐	☐
1.3.4	Ensure that the --service-account-private-key-file argument is set as appropriate	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.3.5	Ensure that the --root-ca-file argument is set as appropriate	☐	☐
1.3.6	Ensure that the RotateKubeletServerCertificate argument is set to true	☐	☐
1.3.7	Ensure that the --bind-address argument is set to 127.0.0.1	☐	☐
1.4.1	Ensure that the --profiling argument is set to false	☐	☐
1.4.2	Ensure that the --bind-address argument is set to 127.0.0.1	☐	☐
2.1	Ensure that the --cert-file and --key-file arguments are set as appropriate	☐	☐
2.3	Ensure that the --auto-tls argument is not set to true	☐	☐
2.4	Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate	☐	☐
2.5	Ensure that the --peer-client-cert-auth argument is set to true	☐	☐
2.6	Ensure that the --peer-auto-tls argument is not set to true	☐	☐
3.1.1	Client certificate authentication should not be used for users	☐	☐
3.1.2	Service account token authentication should not be used for users	☐	☐
3.1.3	Bootstrap token authentication should not be used for users	☐	☐
3.2.1	Ensure that a minimal audit policy is created	☐	☐
4.1.1	Ensure that the kubelet service file permissions are set to 600 or more restrictive	☐	☐
4.1.3	If proxy kubeconfig file exists ensure permissions are set to 600 or more restrictive	☐	☐
4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 or more restrictive	☐	☐
4.1.7	Ensure that the certificate authorities file permissions are set to 600 or more restrictive	☐	☐
4.1.9	If the kubelet config.yaml configuration file is being used validate permissions set to 600 or more restrictive	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
4.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
4.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
4.2.4	Verify that the --read-only-port argument is set to 0	☐	☐
4.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
4.2.8	Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture	☐	☐
4.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
4.2.10	Ensure that the --rotate-certificates argument is not set to false	☐	☐
4.2.11	Verify that the RotateKubeletServerCertificate argument is set to true	☐	☐
4.2.12	Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers	☐	☐
4.3.1	Ensure that the kube-proxy metrics service is bound to localhost	☐	☐
5.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
5.1.5	Ensure that default service accounts are not actively used.	☐	☐
5.2.4	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
5.2.5	Minimize the admission of containers wishing to share the host network namespace	☐	☐
5.2.8	Minimize the admission of containers with the NET_RAW capability	☐	☐
5.2.9	Minimize the admission of containers with added capabilities	☐	☐
5.2.10	Minimize the admission of containers with capabilities assigned	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
5.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	☐	☐
5.7.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	☐	☐
5.7.3	Apply Security Context to Your Pods and Containers	☐	☐

Appendix: CIS Controls v7 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1.1	Ensure that the API server pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.3	Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.5	Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.7	Ensure that the etcd pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.9	Ensure that the Container Network Interface file permissions are set to 600 or more restrictive	☐	☐
1.1.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive	☐	☐
1.1.13	Ensure that the default administrative credential file permissions are set to 600	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.17	Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.19	Ensure that the Kubernetes PKI directory and file ownership is set to root:root	☐	☐
1.1.21	Ensure that the Kubernetes PKI key file permissions are set to 600	☐	☐
1.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
1.2.2	Ensure that the --token-auth-file parameter is not set	☐	☐
1.2.3	Ensure that the DenyServiceExternalIPs is set	☐	☐
1.2.6	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
1.2.7	Ensure that the --authorization-mode argument includes Node	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.2.8	Ensure that the --authorization-mode argument includes RBAC	☐	☐
1.2.9	Ensure that the admission control plugin EventRateLimit is set	☐	☐
1.2.10	Ensure that the admission control plugin AlwaysAdmit is not set	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysPullImages is set	☐	☐
1.2.12	Ensure that the admission control plugin ServiceAccount is set	☐	☐
1.2.13	Ensure that the admission control plugin NamespaceLifecycle is set	☐	☐
1.2.14	Ensure that the admission control plugin NodeRestriction is set	☐	☐
1.2.16	Ensure that the --audit-log-path argument is set	☐	☐
1.2.17	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate	☐	☐
1.2.18	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate	☐	☐
1.2.19	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate	☐	☐
1.2.20	Ensure that the --request-timeout argument is set as appropriate	☐	☐
1.2.22	Ensure that the --service-account-key-file argument is set as appropriate	☐	☐
1.2.23	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate	☐	☐
1.2.24	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
1.2.25	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
1.2.26	Ensure that the --etcd-cafile argument is set as appropriate	☐	☐
1.2.27	Ensure that the --encryption-provider-config argument is set as appropriate	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.2.28	Ensure that encryption providers are appropriately configured	☐	☐
1.2.29	Ensure that the API Server only makes use of Strong Cryptographic Ciphers	☐	☐
1.3.1	Ensure that the --terminated-pod-gc-threshold argument is set as appropriate	☐	☐
1.3.2	Ensure that the --profiling argument is set to false	☐	☐
1.3.3	Ensure that the --use-service-account-credentials argument is set to true	☐	☐
1.3.4	Ensure that the --service-account-private-key-file argument is set as appropriate	☐	☐
1.3.5	Ensure that the --root-ca-file argument is set as appropriate	☐	☐
1.3.6	Ensure that the RotateKubeletServerCertificate argument is set to true	☐	☐
1.3.7	Ensure that the --bind-address argument is set to 127.0.0.1	☐	☐
1.4.1	Ensure that the --profiling argument is set to false	☐	☐
1.4.2	Ensure that the --bind-address argument is set to 127.0.0.1	☐	☐
2.1	Ensure that the --cert-file and --key-file arguments are set as appropriate	☐	☐
2.2	Ensure that the --client-cert-auth argument is set to true	☐	☐
2.3	Ensure that the --auto-tls argument is not set to true	☐	☐
2.4	Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate	☐	☐
2.5	Ensure that the --peer-client-cert-auth argument is set to true	☐	☐
2.6	Ensure that the --peer-auto-tls argument is not set to true	☐	☐
3.1.1	Client certificate authentication should not be used for users	☐	☐
3.1.2	Service account token authentication should not be used for users	☐	☐
3.1.3	Bootstrap token authentication should not be used for users	☐	☐

Recommendation		Set Correctly	
		Yes	No
3.2.1	Ensure that a minimal audit policy is created	☐	☐
3.2.2	Ensure that the audit policy covers key security concerns	☐	☐
4.1.1	Ensure that the kubelet service file permissions are set to 600 or more restrictive	☐	☐
4.1.3	If proxy kubeconfig file exists ensure permissions are set to 600 or more restrictive	☐	☐
4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 or more restrictive	☐	☐
4.1.7	Ensure that the certificate authorities file permissions are set to 600 or more restrictive	☐	☐
4.1.9	If the kubelet config.yaml configuration file is being used validate permissions set to 600 or more restrictive	☐	☐
4.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
4.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
4.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
4.2.4	Verify that the --read-only-port argument is set to 0	☐	☐
4.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
4.2.8	Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture	☐	☐
4.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
4.2.10	Ensure that the --rotate-certificates argument is not set to false	☐	☐
4.2.11	Verify that the RotateKubeletServerCertificate argument is set to true	☐	☐
4.2.12	Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers	☐	☐
4.3.1	Ensure that the kube-proxy metrics service is bound to localhost	☐	☐
5.1.1	Ensure that the cluster-admin role is only used where required	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.1.5	Ensure that default service accounts are not actively used.	☐	☐
5.2.3	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
5.2.4	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
5.2.5	Minimize the admission of containers wishing to share the host network namespace	☐	☐
5.2.8	Minimize the admission of containers with the NET_RAW capability	☐	☐
5.2.9	Minimize the admission of containers with added capabilities	☐	☐
5.2.10	Minimize the admission of containers with capabilities assigned	☐	☐
5.3.1	Ensure that the CNI in use supports Network Policies	☐	☐
5.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
5.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	☐	☐
5.7.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	☐	☐
5.7.3	Apply Security Context to Your Pods and Containers	☐	☐
5.7.4	The default namespace should not be used	☐	☐

Appendix: CIS Controls v7 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.7	Ensure that a unique Certificate Authority is used for etcd	☐	☐
4.2.13	Ensure that a limit is set on pod PIDs	☐	☐
5.2.1	Ensure that the cluster has at least one active policy control mechanism in place	☐	☐
5.2.11	Minimize the admission of Windows HostProcess Containers	☐	☐
5.2.12	Minimize the admission of HostPath volumes	☐	☐
5.2.13	Minimize the admission of containers which use HostPorts	☐	☐

Appendix: CIS Controls v8 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1.1	Ensure that the API server pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.2	Ensure that the API server pod specification file ownership is set to root:root	☐	☐
1.1.3	Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.4	Ensure that the controller manager pod specification file ownership is set to root:root	☐	☐
1.1.5	Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.6	Ensure that the scheduler pod specification file ownership is set to root:root	☐	☐
1.1.7	Ensure that the etcd pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.8	Ensure that the etcd pod specification file ownership is set to root:root	☐	☐
1.1.9	Ensure that the Container Network Interface file permissions are set to 600 or more restrictive	☐	☐
1.1.10	Ensure that the Container Network Interface file ownership is set to root:root	☐	☐
1.1.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive	☐	☐
1.1.12	Ensure that the etcd data directory ownership is set to etcd:etcd	☐	☐
1.1.13	Ensure that the default administrative credential file permissions are set to 600	☐	☐
1.1.14	Ensure that the default administrative credential file ownership is set to root:root	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 600 or more restrictive	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.1.16	Ensure that the scheduler.conf file ownership is set to root:root	☐	☐
1.1.17	Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.18	Ensure that the controller-manager.conf file ownership is set to root:root	☐	☐
1.1.19	Ensure that the Kubernetes PKI directory and file ownership is set to root:root	☐	☐
1.1.20	Ensure that the Kubernetes PKI certificate file permissions are set to 600 or more restrictive	☐	☐
1.1.21	Ensure that the Kubernetes PKI key file permissions are set to 600	☐	☐
1.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
1.2.2	Ensure that the --token-auth-file parameter is not set	☐	☐
1.2.3	Ensure that the DenyServiceExternalIPs is set	☐	☐
1.2.6	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
1.2.7	Ensure that the --authorization-mode argument includes Node	☐	☐
1.2.10	Ensure that the admission control plugin AlwaysAdmit is not set	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysPullImages is set	☐	☐
1.2.12	Ensure that the admission control plugin ServiceAccount is set	☐	☐
1.2.16	Ensure that the --audit-log-path argument is set	☐	☐
1.2.17	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate	☐	☐
1.2.18	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate	☐	☐
1.2.19	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate	☐	☐
1.2.21	Ensure that the --service-account-lookup argument is set to true	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.2.22	Ensure that the --service-account-key-file argument is set as appropriate	☐	☐
1.3.3	Ensure that the --use-service-account-credentials argument is set to true	☐	☐
1.3.4	Ensure that the --service-account-private-key-file argument is set as appropriate	☐	☐
2.5	Ensure that the --peer-client-cert-auth argument is set to true	☐	☐
2.6	Ensure that the --peer-auto-tls argument is not set to true	☐	☐
2.7	Ensure that a unique Certificate Authority is used for etcd	☐	☐
3.1.1	Client certificate authentication should not be used for users	☐	☐
3.1.2	Service account token authentication should not be used for users	☐	☐
3.1.3	Bootstrap token authentication should not be used for users	☐	☐
3.2.1	Ensure that a minimal audit policy is created	☐	☐
4.1.1	Ensure that the kubelet service file permissions are set to 600 or more restrictive	☐	☐
4.1.2	Ensure that the kubelet service file ownership is set to root:root	☐	☐
4.1.3	If proxy kubeconfig file exists ensure permissions are set to 600 or more restrictive	☐	☐
4.1.4	If proxy kubeconfig file exists ensure ownership is set to root:root	☐	☐
4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 or more restrictive	☐	☐
4.1.6	Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root	☐	☐
4.1.7	Ensure that the certificate authorities file permissions are set to 600 or more restrictive	☐	☐
4.1.8	Ensure that the client certificate authorities file ownership is set to root:root	☐	☐
4.1.9	If the kubelet config.yaml configuration file is being used validate permissions set to 600 or more restrictive	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.10	If the kubelet config.yaml configuration file is being used validate file ownership is set to root:root	☐	☐
4.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
4.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
4.2.8	Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture	☐	☐
5.1.5	Ensure that default service accounts are not actively used.	☐	☐
5.1.7	Avoid use of system:masters group	☐	☐
5.1.8	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
5.2.2	Minimize the admission of privileged containers	☐	☐
5.2.6	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
5.2.7	Minimize the admission of root containers	☐	☐
5.3.1	Ensure that the CNI in use supports Network Policies	☐	☐
5.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐

Appendix: CIS Controls v8 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1.1	Ensure that the API server pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.2	Ensure that the API server pod specification file ownership is set to root:root	☐	☐
1.1.3	Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.4	Ensure that the controller manager pod specification file ownership is set to root:root	☐	☐
1.1.5	Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.6	Ensure that the scheduler pod specification file ownership is set to root:root	☐	☐
1.1.7	Ensure that the etcd pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.8	Ensure that the etcd pod specification file ownership is set to root:root	☐	☐
1.1.9	Ensure that the Container Network Interface file permissions are set to 600 or more restrictive	☐	☐
1.1.10	Ensure that the Container Network Interface file ownership is set to root:root	☐	☐
1.1.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive	☐	☐
1.1.12	Ensure that the etcd data directory ownership is set to etcd:etcd	☐	☐
1.1.13	Ensure that the default administrative credential file permissions are set to 600	☐	☐
1.1.14	Ensure that the default administrative credential file ownership is set to root:root	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 600 or more restrictive	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.1.16	Ensure that the scheduler.conf file ownership is set to root:root	☐	☐
1.1.17	Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.18	Ensure that the controller-manager.conf file ownership is set to root:root	☐	☐
1.1.19	Ensure that the Kubernetes PKI directory and file ownership is set to root:root	☐	☐
1.1.20	Ensure that the Kubernetes PKI certificate file permissions are set to 600 or more restrictive	☐	☐
1.1.21	Ensure that the Kubernetes PKI key file permissions are set to 600	☐	☐
1.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
1.2.2	Ensure that the --token-auth-file parameter is not set	☐	☐
1.2.3	Ensure that the DenyServiceExternalIPs is set	☐	☐
1.2.4	Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate	☐	☐
1.2.5	Ensure that the --kubelet-certificate-authority argument is set as appropriate	☐	☐
1.2.6	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
1.2.7	Ensure that the --authorization-mode argument includes Node	☐	☐
1.2.9	Ensure that the admission control plugin EventRateLimit is set	☐	☐
1.2.10	Ensure that the admission control plugin AlwaysAdmit is not set	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysPullImages is set	☐	☐
1.2.12	Ensure that the admission control plugin ServiceAccount is set	☐	☐
1.2.16	Ensure that the --audit-log-path argument is set	☐	☐
1.2.17	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.2.18	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate	☐	☐
1.2.19	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate	☐	☐
1.2.21	Ensure that the --service-account-lookup argument is set to true	☐	☐
1.2.22	Ensure that the --service-account-key-file argument is set as appropriate	☐	☐
1.2.23	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate	☐	☐
1.2.24	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
1.2.25	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
1.2.26	Ensure that the --etcd-cafile argument is set as appropriate	☐	☐
1.2.27	Ensure that the --encryption-provider-config argument is set as appropriate	☐	☐
1.2.28	Ensure that encryption providers are appropriately configured	☐	☐
1.2.29	Ensure that the API Server only makes use of Strong Cryptographic Ciphers	☐	☐
1.3.1	Ensure that the --terminated-pod-gc-threshold argument is set as appropriate	☐	☐
1.3.2	Ensure that the --profiling argument is set to false	☐	☐
1.3.3	Ensure that the --use-service-account-credentials argument is set to true	☐	☐
1.3.4	Ensure that the --service-account-private-key-file argument is set as appropriate	☐	☐
1.3.5	Ensure that the --root-ca-file argument is set as appropriate	☐	☐
1.3.6	Ensure that the RotateKubeletServerCertificate argument is set to true	☐	☐
1.3.7	Ensure that the --bind-address argument is set to 127.0.0.1	☐	☐
1.4.1	Ensure that the --profiling argument is set to false	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.4.2	Ensure that the --bind-address argument is set to 127.0.0.1	☐	☐
2.1	Ensure that the --cert-file and --key-file arguments are set as appropriate	☐	☐
2.2	Ensure that the --client-cert-auth argument is set to true	☐	☐
2.3	Ensure that the --auto-tls argument is not set to true	☐	☐
2.4	Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate	☐	☐
2.5	Ensure that the --peer-client-cert-auth argument is set to true	☐	☐
2.6	Ensure that the --peer-auto-tls argument is not set to true	☐	☐
2.7	Ensure that a unique Certificate Authority is used for etcd	☐	☐
3.1.1	Client certificate authentication should not be used for users	☐	☐
3.1.2	Service account token authentication should not be used for users	☐	☐
3.1.3	Bootstrap token authentication should not be used for users	☐	☐
3.2.1	Ensure that a minimal audit policy is created	☐	☐
3.2.2	Ensure that the audit policy covers key security concerns	☐	☐
4.1.1	Ensure that the kubelet service file permissions are set to 600 or more restrictive	☐	☐
4.1.2	Ensure that the kubelet service file ownership is set to root:root	☐	☐
4.1.3	If proxy kubeconfig file exists ensure permissions are set to 600 or more restrictive	☐	☐
4.1.4	If proxy kubeconfig file exists ensure ownership is set to root:root	☐	☐
4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 or more restrictive	☐	☐
4.1.6	Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root	☐	☐
4.1.7	Ensure that the certificate authorities file permissions are set to 600 or more restrictive	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.8	Ensure that the client certificate authorities file ownership is set to root:root	☐	☐
4.1.9	If the kubelet config.yaml configuration file is being used validate permissions set to 600 or more restrictive	☐	☐
4.1.10	If the kubelet config.yaml configuration file is being used validate file ownership is set to root:root	☐	☐
4.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
4.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
4.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
4.2.4	Verify that the --read-only-port argument is set to 0	☐	☐
4.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
4.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
4.2.8	Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture	☐	☐
4.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
4.2.10	Ensure that the --rotate-certificates argument is not set to false	☐	☐
4.2.11	Verify that the RotateKubeletServerCertificate argument is set to true	☐	☐
4.2.12	Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers	☐	☐
4.3.1	Ensure that the kube-proxy metrics service is bound to localhost	☐	☐
5.1.5	Ensure that default service accounts are not actively used.	☐	☐
5.1.7	Avoid use of system:masters group	☐	☐
5.1.8	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
5.2.2	Minimize the admission of privileged containers	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.2.4	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
5.2.5	Minimize the admission of containers wishing to share the host network namespace	☐	☐
5.2.6	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
5.2.7	Minimize the admission of root containers	☐	☐
5.2.8	Minimize the admission of containers with the NET_RAW capability	☐	☐
5.2.9	Minimize the admission of containers with added capabilities	☐	☐
5.2.10	Minimize the admission of containers with capabilities assigned	☐	☐
5.3.1	Ensure that the CNI in use supports Network Policies	☐	☐
5.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
5.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	☐	☐
5.7.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	☐	☐
5.7.4	The default namespace should not be used	☐	☐

Appendix: CIS Controls v8 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
1.1.1	Ensure that the API server pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.2	Ensure that the API server pod specification file ownership is set to root:root	☐	☐
1.1.3	Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.4	Ensure that the controller manager pod specification file ownership is set to root:root	☐	☐
1.1.5	Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.6	Ensure that the scheduler pod specification file ownership is set to root:root	☐	☐
1.1.7	Ensure that the etcd pod specification file permissions are set to 600 or more restrictive	☐	☐
1.1.8	Ensure that the etcd pod specification file ownership is set to root:root	☐	☐
1.1.9	Ensure that the Container Network Interface file permissions are set to 600 or more restrictive	☐	☐
1.1.10	Ensure that the Container Network Interface file ownership is set to root:root	☐	☐
1.1.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive	☐	☐
1.1.12	Ensure that the etcd data directory ownership is set to etcd:etcd	☐	☐
1.1.13	Ensure that the default administrative credential file permissions are set to 600	☐	☐
1.1.14	Ensure that the default administrative credential file ownership is set to root:root	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 600 or more restrictive	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.1.16	Ensure that the scheduler.conf file ownership is set to root:root	☐	☐
1.1.17	Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive	☐	☐
1.1.18	Ensure that the controller-manager.conf file ownership is set to root:root	☐	☐
1.1.19	Ensure that the Kubernetes PKI directory and file ownership is set to root:root	☐	☐
1.1.20	Ensure that the Kubernetes PKI certificate file permissions are set to 600 or more restrictive	☐	☐
1.1.21	Ensure that the Kubernetes PKI key file permissions are set to 600	☐	☐
1.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
1.2.2	Ensure that the --token-auth-file parameter is not set	☐	☐
1.2.3	Ensure that the DenyServiceExternalIPs is set	☐	☐
1.2.4	Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate	☐	☐
1.2.5	Ensure that the --kubelet-certificate-authority argument is set as appropriate	☐	☐
1.2.6	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
1.2.7	Ensure that the --authorization-mode argument includes Node	☐	☐
1.2.8	Ensure that the --authorization-mode argument includes RBAC	☐	☐
1.2.9	Ensure that the admission control plugin EventRateLimit is set	☐	☐
1.2.10	Ensure that the admission control plugin AlwaysAdmit is not set	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysPullImages is set	☐	☐
1.2.12	Ensure that the admission control plugin ServiceAccount is set	☐	☐
1.2.14	Ensure that the admission control plugin NodeRestriction is set	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.2.16	Ensure that the --audit-log-path argument is set	☐	☐
1.2.17	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate	☐	☐
1.2.18	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate	☐	☐
1.2.19	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate	☐	☐
1.2.21	Ensure that the --service-account-lookup argument is set to true	☐	☐
1.2.22	Ensure that the --service-account-key-file argument is set as appropriate	☐	☐
1.2.23	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate	☐	☐
1.2.24	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
1.2.25	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
1.2.26	Ensure that the --etcd-cafile argument is set as appropriate	☐	☐
1.2.27	Ensure that the --encryption-provider-config argument is set as appropriate	☐	☐
1.2.28	Ensure that encryption providers are appropriately configured	☐	☐
1.2.29	Ensure that the API Server only makes use of Strong Cryptographic Ciphers	☐	☐
1.3.1	Ensure that the --terminated-pod-gc-threshold argument is set as appropriate	☐	☐
1.3.2	Ensure that the --profiling argument is set to false	☐	☐
1.3.3	Ensure that the --use-service-account-credentials argument is set to true	☐	☐
1.3.4	Ensure that the --service-account-private-key-file argument is set as appropriate	☐	☐
1.3.5	Ensure that the --root-ca-file argument is set as appropriate	☐	☐
1.3.6	Ensure that the RotateKubeletServerCertificate argument is set to true	☐	☐

Recommendation		Set Correctly	
		Yes	No
1.3.7	Ensure that the --bind-address argument is set to 127.0.0.1	☐	☐
1.4.1	Ensure that the --profiling argument is set to false	☐	☐
1.4.2	Ensure that the --bind-address argument is set to 127.0.0.1	☐	☐
2.1	Ensure that the --cert-file and --key-file arguments are set as appropriate	☐	☐
2.2	Ensure that the --client-cert-auth argument is set to true	☐	☐
2.3	Ensure that the --auto-tls argument is not set to true	☐	☐
2.4	Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate	☐	☐
2.5	Ensure that the --peer-client-cert-auth argument is set to true	☐	☐
2.6	Ensure that the --peer-auto-tls argument is not set to true	☐	☐
2.7	Ensure that a unique Certificate Authority is used for etcd	☐	☐
3.1.1	Client certificate authentication should not be used for users	☐	☐
3.1.2	Service account token authentication should not be used for users	☐	☐
3.1.3	Bootstrap token authentication should not be used for users	☐	☐
3.2.1	Ensure that a minimal audit policy is created	☐	☐
3.2.2	Ensure that the audit policy covers key security concerns	☐	☐
4.1.1	Ensure that the kubelet service file permissions are set to 600 or more restrictive	☐	☐
4.1.2	Ensure that the kubelet service file ownership is set to root:root	☐	☐
4.1.3	If proxy kubeconfig file exists ensure permissions are set to 600 or more restrictive	☐	☐
4.1.4	If proxy kubeconfig file exists ensure ownership is set to root:root	☐	☐
4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 600 or more restrictive	☐	☐
4.1.6	Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.7	Ensure that the certificate authorities file permissions are set to 600 or more restrictive	☐	☐
4.1.8	Ensure that the client certificate authorities file ownership is set to root:root	☐	☐
4.1.9	If the kubelet config.yaml configuration file is being used validate permissions set to 600 or more restrictive	☐	☐
4.1.10	If the kubelet config.yaml configuration file is being used validate file ownership is set to root:root	☐	☐
4.2.1	Ensure that the --anonymous-auth argument is set to false	☐	☐
4.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
4.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
4.2.4	Verify that the --read-only-port argument is set to 0	☐	☐
4.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
4.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
4.2.8	Ensure that the eventRecordQPS argument is set to a level which ensures appropriate event capture	☐	☐
4.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	☐	☐
4.2.10	Ensure that the --rotate-certificates argument is not set to false	☐	☐
4.2.11	Verify that the RotateKubeletServerCertificate argument is set to true	☐	☐
4.2.12	Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers	☐	☐
4.3.1	Ensure that the kube-proxy metrics service is bound to localhost	☐	☐
5.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
5.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
5.1.4	Minimize access to create pods	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.1.5	Ensure that default service accounts are not actively used.	☐	☐
5.1.7	Avoid use of system:masters group	☐	☐
5.1.8	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
5.1.9	Minimize access to create persistent volumes	☐	☐
5.1.10	Minimize access to the proxy sub-resource of nodes	☐	☐
5.1.11	Minimize access to the approval sub-resource of certificatesigningrequests objects	☐	☐
5.1.12	Minimize access to webhook configuration objects	☐	☐
5.1.13	Minimize access to the service account token creation	☐	☐
5.2.2	Minimize the admission of privileged containers	☐	☐
5.2.3	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
5.2.4	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
5.2.5	Minimize the admission of containers wishing to share the host network namespace	☐	☐
5.2.6	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
5.2.7	Minimize the admission of root containers	☐	☐
5.2.8	Minimize the admission of containers with the NET_RAW capability	☐	☐
5.2.9	Minimize the admission of containers with added capabilities	☐	☐
5.2.10	Minimize the admission of containers with capabilities assigned	☐	☐
5.3.1	Ensure that the CNI in use supports Network Policies	☐	☐
5.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
5.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	☐	☐
5.7.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	☐	☐
5.7.4	The default namespace should not be used	☐	☐

Appendix: CIS Controls v8 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
4.2.13	Ensure that a limit is set on pod PIDs	☐	☐
5.2.1	Ensure that the cluster has at least one active policy control mechanism in place	☐	☐
5.2.11	Minimize the admission of Windows HostProcess Containers	☐	☐
5.2.12	Minimize the admission of HostPath volumes	☐	☐
5.2.13	Minimize the admission of containers which use HostPorts	☐	☐

Appendix: Change History

Date	Version	Changes for this version
2/27/2024	1.9.0	Validated all AAC against Kubernetes v1.29
2/17/2024	1.9.0	Added CIS-CAT automated Audit Check for recommendation 5.1.1
2/17/2024	1.9.0	Added CIS-CAT automated Audit Check for recommendation 5.1.2
2/17/2024	1.9.0	Added CIS-CAT automated Audit Check for recommendation 5.1.3
2/17/2024	1.9.0	Added CIS-CAT automated Audit Check for recommendation 5.1.4
2/17/2024	1.9.0	Added CIS-CAT automated Audit Check for recommendation 5.1.5
2/17/2024	1.9.0	Added CIS-CAT automated Audit Check for recommendation 5.1.6
1/18/2023	1.9.0	Validated all AAC against Kubernetes v1.28
9/15/23	1.8.0	Validated all AAC against Kubernetes v1.27
9/1/23	1.8.0	Ticket #19171 Improved the audit process for ensuring a cluster has at least one active policy control mechanism

Date	Version	Changes for this version
8/25/23	1.8.0	Ticket #18657 Edited recommendations to use %U;%G vs a% when retrieving file ownership.
8/25/23	1.8.0	Ticket #18656 Updated recommendations to improve the audit command to retrieve file ownership.
6/25/23	1.8.0	Ticket #18649 Updated recommendations to assist uders w/ numeric mode & octal mode
4/15/2023	1.7.1	Modified AAC for Profile Level 2 to fix executable bugs
2/7/2023	1.7.0	Ticket #16492 Reformatted audit Procedure for recommendation 4.2.8
2/7/2023	1.7.0	Ticket #16491 Edited recommendation 5.2.10
2/7/2023	1.7.0	Ticket # 16607 Updated Recommendation 1.1.19 to provide a consistent audit.
2/18/2023	1.7.0	Ticket # 17377 Removed recommendation to set –protected-kernel-defaults
2/19/2023	1.7.0	Ticket # 17609

Date	Version	Changes for this version
		Updated the default section of recommendation 1.2.3
2/19/2023	1.7.0	Ticket # 16624 Updated Recommendation 5.3.2 moved flags in audit process to end of the command line
2/20/2023	1.7.0	Ticket # 16625Updated Recommendation 5.7.1 default value statement.
2/20/2023	1.7.0	
2/20/2023	1.7.0	