CIS Benchmarks

CIS Kubernetes Benchmark

v1.6.1 - 10-01-2020

Terms of Use

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

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

Table of Contents

Overview

This document provides prescriptive guidance for establishing a secure configuration posture for Kubernetes 1.16 - 1.18. 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 1.16 - 1.18.

Consensus Guidance

This benchmark was created using a consensus review process comprised of subject matter experts. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

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

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
Monospace font	Used for inline code, commands, or examples. Text should be interpreted exactly as presented.
<italic font in brackets>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

Assessment Status

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

Automated

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

Manual

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

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1 - 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
Randall Mowen (GISP)

Contributors
Pravin Goyal
Prabhu Angadi Security Content Author (Compliance | Configuration | Checklist)
Jordan Liggitt
Eric Chiang
Jordan Rakoske (GSEC, GCWN)
Sara Archacki
Maya Kaczorowski
Andrew Martin
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 Master Node Configuration Files

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

Impact:

None

Audit:

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

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

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

Remediation:

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

```
chmod 644 /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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.2 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 master 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 master 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

The controller manager pod specification file controls various parameters that set the behavior of the Controller Manager on the master node. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None

Audit:

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

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

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

Remediation:

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

```
chmod 644 /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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.4 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 master 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 master 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

The scheduler pod specification file controls various parameters that set the behavior of the Scheduler service in the master node. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None

Audit:

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

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

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

Remediation:

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

```
chmod 644 /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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.6 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 master 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 master 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

Impact:

None

Audit:

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

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

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

Remediation:

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

```
chmod 644 /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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.8 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 master 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 master 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

Impact:

None

Audit:

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

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

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

Remediation:

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

```
chmod 644 <path/to/cni/files>
```

Default Value:

NA

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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 master 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 master 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.11 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:

Version 6

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

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:

Version 6

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

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

The `admin.conf` is the administrator kubeconfig file defining various settings for the administration of the cluster. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None.

Audit:

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

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

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

Remediation:

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

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

Default Value:

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

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

Impact:

None.

Audit:

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

```
stat -c %U:%G /etc/kubernetes/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 master node. For example,

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

Default Value:

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

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

The `scheduler.conf` file is the kubeconfig file for the Scheduler. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None

Audit:

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

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

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

Remediation:

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

```
chmod 644 /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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.16 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 master 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 master 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:

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

The `controller-manager.conf` file is the kubeconfig file for the Controller Manager. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None

Audit:

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

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

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

Remediation:

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

```
chmod 644 /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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.18 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 master 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 master 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.19 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 master 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 master 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that Kubernetes PKI certificate files have permissions of `644` 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 `644` or more restrictive to protect their integrity.

Impact:

None

Audit:

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

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

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

Remediation:

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

```
chmod -R 644 /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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.1.21 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 master node. For example,

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

Verify that the permissions are `600`.

Remediation:

Run the below command (based on the file location on your system) on the master 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

1.2 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 master node:

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

Verify that the `--anonymous-auth` argument is set to `false`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master 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:

Version 6

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

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

Profile Applicability:

- Level 1 - Master Node

Description:

Do not use basic authentication.

Rationale:

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

Impact:

You will have to configure and use alternate authentication mechanisms such as tokens and certificates. Username and password for basic authentication could no longer be used.

Audit:

Run the following command on the master node:

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

Verify that the `--basic-auth-file` argument does not exist.

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 `--basic-auth-file=<filename>` parameter.

Default Value:

By default, basic authentication is not set.

References:

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

CIS Controls:

Version 6

16.14 Encrypt/Hash All Authentication Files And Monitor Their Access

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

Version 7

16.4 Encrypt or Hash all Authentication Credentials

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

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

Profile Applicability:

- Level 1 - 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 master node:

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

Verify that the `--token-auth-file` argument does not exist.

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:

Version 6

16.14 Encrypt/Hash All Authentication Files And Monitor Their Access

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

Version 7

16.4 Encrypt or Hash all Authentication Credentials

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

1.2.4 Ensure that the --kubelet-https argument is set to true (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Use https for kubelet connections.

Rationale:

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

Impact:

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

Audit:

Run the following command on the master node:

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

Verify that the `--kubelet-https` argument either does not exist or is set to `true`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and remove the `--kubelet-https` parameter.

Default Value:

By default, kubelet connections are over https.

References:

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

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

1.2.5 Ensure that the --kubelet-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 master node:

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

Verify that the `--kubelet-client-certificate` and `--kubelet-client-key` arguments exist and they are set as appropriate.

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 master 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:

Version 6

3.4 Use Only Secure Channels For Remote System Administration

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

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

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

1.2.6 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 master node:

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

Verify that the `--kubelet-certificate-authority` argument exists and is set as appropriate.

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 master 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:

Version 6

3.4 Use Only Secure Channels For Remote System Administration

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

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

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

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

Profile Applicability:

- Level 1 - 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 master 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 master 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:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

1.2.8 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 master 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 master 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:

Version 6

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

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running
Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

1.2.9 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 master 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 master 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:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

1.2.10 Ensure that the 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.

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

Impact:

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

Audit:

Run the following command on the master 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:

Version 6

8.4 Enable Anti-exploitation Features (i.e. DEP, ASLR, EMET)

Enable anti-exploitation features such as Data Execution Prevention (DEP), Address Space Layout Randomization (ASLR), virtualization/containerization, etc. For increased protection, deploy capabilities such as Enhanced Mitigation Experience Toolkit (EMET) that can be configured to apply these protections to a broader set of applications and executables.

Version 7

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

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

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

Profile Applicability:

- Level 1 - 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 master 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 master 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:

Version 6

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

1.2.12 Ensure that the admission control plugin AlwaysPullImages is 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 master 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 master 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:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

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

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

Impact:

This admission controller should only be used where Pod Security Policies cannot be used on the cluster, as it can interact poorly with certain Pod Security Policies

Audit:

Run the following command on the master node:

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

Verify that the `--enable-admission-plugins` argument is set to a value that includes `SecurityContextDeny`, if `PodSecurityPolicy` is not included.

Remediation:

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 include `SecurityContextDeny`, unless `PodSecurityPolicy` is already in place.

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

Default Value:

By default, `SecurityContextDeny` is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers/#securitycontextdeny
3. https://kubernetes.io/docs/user-guide/pod-security-policy/#working-with-rbac

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - 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 master 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:

Version 6

16 Account Monitoring and Control
Account Monitoring and Control

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

Profile Applicability:

- Level 1 - 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 master 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 master 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:

Version 6

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

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

Profile Applicability:

- Level 1 - Master Node

Description:

Reject creating pods that do not match Pod Security Policies.

Rationale:

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

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

Impact:

The policy objects must be created and granted before pod creation would be allowed.

Audit:

Run the following command on the master node:

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

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

Remediation:

Follow the documentation and create Pod Security Policy objects as per your environment. 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 `PodSecurityPolicy`:

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

Then restart the API Server.

Default Value:

By default, `PodSecurityPolicy` is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers/#podsecuritypolicy
3. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies

CIS Controls:

Version 6

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

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

Profile Applicability:

- Level 1 - 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 master 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:

Version 6

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

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

Profile Applicability:

- Level 1 - Master Node

Description:

Do not bind the insecure API service.

Rationale:

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

Impact:

Connections to the API server will require valid authentication credentials.

Audit:

Run the following command on the master node:

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

Verify that the `--insecure-bind-address` argument does not exist.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and remove the `--insecure-bind-address` parameter.

Default Value:

By default, the insecure bind address is not set.

References:

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

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

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

Profile Applicability:

- Level 1 - Master Node

Description:

Do not bind to insecure port.

Rationale:

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

Impact:

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

This includes:

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

Audit:

Run the following command on the master node:

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

Verify that the `--insecure-port` argument is set to `0`.

Remediation:

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

```
--insecure-port=0
```

Default Value:

By default, the insecure port is set to 8080.

References:

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

CIS Controls:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

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

Profile Applicability:

- Level 1 - Master Node

Description:

Do not disable the secure port.

Rationale:

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

Impact:

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

Audit:

Run the following command on the master node:

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

Verify that the `--secure-port` argument is either not set or is set to an integer value between 1 and 65535.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and either remove the `--secure-port` parameter or set it to a different (non-zero) desired port.

Default Value:

By default, port 6443 is used as the secure port.

References:

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

CIS Controls:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

1.2.21 Ensure that the --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 master 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 master 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:

Version 6

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

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

Profile Applicability:

- Level 1 - 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 master 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 master 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:

Version 6

6.2 Ensure Audit Log Settings Support Appropriate Log Entry Formatting

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

Version 7

6.2 Activate audit logging

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

6.3 Enable Detailed Logging

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

1.2.23 Ensure that the --audit-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 master 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 master 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:

Version 6

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

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

Version 7

6.4 Ensure adequate storage for logs

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

1.2.24 Ensure that the --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 master 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 master 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:

Version 6

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

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

Version 7

6.4 Ensure adequate storage for logs

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

1.2.25 Ensure that the --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 master 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 master 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:

Version 6

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

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

Version 7

6.4 Ensure adequate storage for logs

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

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

Profile Applicability:

- Level 1 - 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 master 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:

Version 6

14.6 Enforce Detailed Audit Logging For Sensitive Information

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

Version 7

14.9 Enforce Detail Logging for Access or Changes to Sensitive Data

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

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

Profile Applicability:

- Level 1 - 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 master 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 master 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:

Version 6

16 Account Monitoring and Control
Account Monitoring and Control

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

Profile Applicability:

- Level 1 - 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 master 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 master 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:

Version 6

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

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

1.2.29 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 master 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:

Version 6

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

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

Profile Applicability:

- Level 1 - 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 master 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` arguments are 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:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

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

Profile Applicability:

- Level 1 - 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 master 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:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

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

Profile Applicability:

- Level 1 - 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 master 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:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

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

Profile Applicability:

- Level 1 - 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 master 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:

Version 6

14.5 Encrypt At Rest Sensitive Information

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

Version 7

14.8 Encrypt Sensitive Information at Rest

Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.

1.2.34 Ensure that encryption providers are appropriately configured (Manual)

Profile Applicability:

- Level 1 - 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 master 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:

Version 6

14.5 Encrypt At Rest Sensitive Information

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

Version 7

14.8 Encrypt Sensitive Information at Rest

Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.

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

Profile Applicability:

- Level 1 - 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 master 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 master node and set the below parameter.

```
--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
```

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:

Version 6

3.4 Use Only Secure Channels For Remote System Administration

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

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

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

1.3 Controller Manager

This section contains recommendations relating to Controller Manager configuration flags

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 master 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 master 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:

Version 6

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

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 master 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 master 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:

Version 6

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

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 master 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 master 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:

Version 6

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

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 master 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 master 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:

Version 6

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 master 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 master 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:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

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

Profile Applicability:

- Level 2 - 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 master 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 master 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:

Version 6

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

Version 7

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

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

Profile Applicability:

- Level 1 - 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 master 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 master 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:

Version 6

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

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running
Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

1.4 Scheduler

This section contains recommendations relating to Scheduler configuration flags

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 master 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 master 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:

Version 6

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

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 master 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 master 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:

Version 6

9.1 Limit Open Ports, Protocols, and Services

Ensure that only ports, protocols, and services with validated business needs are running on each system.

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running

Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

2 etcd

This section covers recommendations for etcd configuration.

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:

Version 6

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

Version 7

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

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

Profile Applicability:

- Level 1 - 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:

Version 6

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

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:

Version 6

14.2 Encrypt All Sensitive Information Over Less-trusted Networks

All communication of sensitive information over less-trusted networks should be encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

Version 7

14.4 Encrypt All Sensitive Information in Transit

Encrypt all sensitive information in transit.

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

Profile Applicability:

- Level 1 - 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:

Version 6

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

Version 7

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

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

Profile Applicability:

- Level 1 - 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:

Version 6

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

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

Profile Applicability:

- Level 1 - 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:

Version 6

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

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:

Version 6

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

3 Control Plane Configuration

This section contains recommendations for cluster-wide areas, such as authentication and logging. Unlike section 1 these recommendations should apply to all deployments.

3.1 Authentication and Authorization

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

Profile Applicability:

- Level 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.

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:

Version 7

6.2 Activate audit logging

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

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

Profile Applicability:

- Level 2 - 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:

Version 6

14.6 Enforce Detailed Audit Logging For Sensitive Information

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

Version 7

14.9 Enforce Detail Logging for Access or Changes to Sensitive Data

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

4 Worker Nodes

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

Note that these components may also run on Kubernetes master nodes, so the recommendations in this section should be applied to master nodes as well as worker nodes where the master nodes make use of these components.

4.1 Worker Node Configuration Files

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

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 644 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Worker Node

Description:

Ensure that the `kubelet` service file has permissions of `644` or more restrictive.

Rationale:

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

Impact:

None

Audit:

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

```
export kubelet_service_config=/etc/systemd/system/kubelet.service.d/10-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 `644` or more restrictive.

Remediation:

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

```
chmod 755 /etc/systemd/system/kubelet.service.d/10-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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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 / of the kubelet service config file.
Please set $kubelet_service_config= based on the file location on your system
for example:

```
export kubelet_service_config=/etc/systemd/system/kubelet.service.d/10-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 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/10-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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

4.1.3 If proxy kubeconfig file exists ensure permissions are set to 644 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 `644` or more restrictive.

Rationale:

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

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

Impact:

None

Audit:

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

```

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

Remediation:

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

```
chmod 644 <proxy kubeconfig file>
```

Default Value:

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

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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 %a
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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

4.1.5 Ensure that the --kubeconfig kubelet.conf file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1 - Worker Node

Description:

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

Rationale:

The `kubelet.conf` file is the kubeconfig file for the node, and controls various parameters that set the behavior and identity of the worker node. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None

Audit:

Automated AAC auditing has been modified to allow CIS-CAT to input a variable for the / of the kubelet config file.
Please set $kubelet_config= 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/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Verify that the ownership is set to `root:root`.Verify that the permissions are `644` or more restrictive.

Remediation:

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

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

Default Value:

By default, `kubelet.conf` file has permissions of `640`.

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

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 / of the kubelet config file.
Please set $kubelet_config= 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/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/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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

4.1.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Manual)

Profile Applicability:

- Level 1 - Worker Node

Description:

Ensure that the certificate authorities file has permissions of `644` or more restrictive.

Rationale:

The certificate authorities file controls the authorities used to validate API requests. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None

Audit:

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 644 <filename>
```

Default Value:

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

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

14.4 Protect Information With Access Control Lists

All information stored on systems shall be protected with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

Version 7

14.6 Protect Information through Access Control Lists

Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.

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

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

4.1.9 Ensure that the kubelet --config configuration file has permissions set to 644 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 644 or more restrictive.

Rationale:

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

Impact:

None

Audit:

Automated AAC auditing has been modified to allow CIS-CAT to input a variable for the / of the kubelet config yaml file.
Please set $kubelet_config_yaml= 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 `644` or more restrictive.

Remediation:

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

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

Default Value:

By default, the /var/lib/kubelet/config.yaml file as set up by `kubeadm` has permissions of 644.

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

4.1.10 Ensure that the kubelet --config configuration 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 / of the kubelet config yaml file.
Please set $kubelet_config_yaml= 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 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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/systemd/system/kubelet.service.d/10-kubeadm.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:

Version 6

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

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

Profile Applicability:

- Level 1 - 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/systemd/system/kubelet.service.d/10-kubeadm.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:

Version 6

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

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

Profile Applicability:

- Level 1 - 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/systemd/system/kubelet.service.d/10-kubeadm.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:

Version 6

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

Version 7

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

4.2.4 Verify that the --read-only-port 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/systemd/system/kubelet.service.d/10-kubeadm.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:

Version 6

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

Version 7

9.2 Ensure Only Approved Ports, Protocols and Services Are Running
Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.

4.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (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/systemd/system/kubelet.service.d/10-kubeadm.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:

Version 6

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

4.2.6 Ensure that the --protect-kernel-defaults argument is set to true (Automated)

Profile Applicability:

- Level 1 - Worker Node

Description:

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

Rationale:

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

Impact:

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

Audit:

Run the following command on each node:

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

Verify that the `--protect-kernel-defaults` argument is set to `true`.
If the `--protect-kernel-defaults` argument is not present, check that there is a Kubelet config file specified by `--config`, and that the file sets `protectKernelDefaults` to `true`.

Remediation:

If using a Kubelet config file, edit the file to set `protectKernelDefaults: true`.
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.

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

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

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

Default Value:

By default, `--protect-kernel-defaults` is not set.

References:

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

CIS Controls:

Version 6

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

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

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

Profile Applicability:

- Level 1 - 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/systemd/system/kubelet.service.d/10-kubeadm.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:

Version 6

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

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

Profile Applicability:

- Level 1 - 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:

Version 6

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

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

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

Profile Applicability:

- Level 2 - Worker Node

Description:

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

Rationale:

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

Impact:

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

Audit:

Run the following command on each node:

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

Review the value set for the `--event-qps` argument and determine whether this has been set to an appropriate level for the cluster. The value of `0` can be used to ensure that all events are captured.
If the `--event-qps` 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, `--event-qps` 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:

Version 6

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

4.2.10 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (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/systemd/system/kubelet.service.d/10-kubeadm.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
```

4.2.11 Ensure that the --rotate-certificates argument is not set to false (Manual)

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/systemd/system/kubelet.service.d/10-kubeadm.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:

Version 6

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

Version 7

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

4.2.12 Verify that the RotateKubeletServerCertificate argument is set to true (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/systemd/system/kubelet.service.d/10-kubeadm.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 disabled.

References:

1. https://github.com/kubernetes/kubernetes/pull/45059
2. https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/#kubelet-configuration

CIS Controls:

Version 6

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

Version 7

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

4.2.13 Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)

Profile Applicability:

- Level 1 - 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/systemd/system/kubelet.service.d/10-kubeadm.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:

Version 6

3.4 Use Only Secure Channels For Remote System Administration
Perform all remote administration of servers, workstation, network devices, and similar

equipment over secure channels. Protocols such as telnet, VNC, RDP, or others that do not actively support strong encryption should only be used if they are performed over a secondary encryption channel, such as SSL, TLS or IPSEC.

Version 7

4.5 Use Multifactor Authentication For All Administrative Access

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

5 Policies

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

5.1 RBAC and Service Accounts

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

Profile Applicability:

- Level 1 - 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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

5.1.2 Minimize access to secrets (Manual)

Profile Applicability:

- Level 1 - 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
```

5.1.3 Minimize wildcard use in Roles and ClusterRoles (Manual)

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.

5.1.4 Minimize access to create pods (Manual)

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
```

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

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/

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

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:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

5.2 Pod Security Policies

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

```
kubectl get psp
```

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

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

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

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

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

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

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

5.2.3 Minimize the admission of containers wishing to share the host IPC namespace (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 PodSecurityPolicy (PSP) defined which does not permit containers to share the host IPC namespace.

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

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

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

5.2.4 Minimize the admission of containers wishing to share the host network namespace (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 PodSecurityPolicy (PSP) defined which does not permit containers to share the host network namespace.

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

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

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

5.2.5 Minimize the admission of containers with allowPrivilegeEscalation (Manual)

Profile Applicability:

- Level 1 - Master Node

Description:

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

Rationale:

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

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

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether privileged is enabled:

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

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

5.2.6 Minimize the admission of root containers (Manual)

Profile Applicability:

- Level 2 - 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 PodSecurityPolicy (PSP) defined which does not permit root users in a container.

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

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

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

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

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

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 1 - 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 PodSecurityPolicy (PSP) defined which prevents containers with the NET_RAW capability from launching.

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether NET_RAW is disabled:

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

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

1. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

5.2.8 Minimize the admission of containers with added capabilities (Manual)

Profile Applicability:

- Level 1 - 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 PodSecurityPolicy (PSP) defined which prevents containers with capabilities beyond the default set from launching.

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

Impact:

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

Audit:

Get the set of PSPs with the following command:

```
kubectl get psp
```

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

1. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

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

Profile Applicability:

- Level 2 - 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:

Get the set of PSPs with the following command:

```
kubectl get psp
```

For each PSP, check whether capabilities have been forbidden:

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

Remediation:

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

Default Value:

By default, PodSecurityPolicies are not defined.

References:

1. https://kubernetes.io/docs/concepts/policy/pod-security-policy/#enabling-pod-security-policies
2. https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/

CIS Controls:

Version 6

5.1 Minimize And Sparingly Use Administrative Privileges

Minimize administrative privileges and only use administrative accounts when they are required. Implement focused auditing on the use of administrative privileged functions and monitor for anomalous behavior.

5.3 Network Policies and CNI

5.3.1 Ensure that the CNI in use supports Network Policies (Manual)

Profile Applicability:

- Level 1 - 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.

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 --all-namespaces get networkpolicy
```

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

Remediation:

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

Default Value:

By default, network policies are not created.

References:

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

CIS Controls:

Version 6

14.1 Implement Network Segmentation Based On Information Class

Segment the network based on the label or classification level of the information stored on the servers. Locate all sensitive information on separated VLANS with firewall filtering to ensure that only authorized individuals are only able to communicate with systems necessary to fulfill their specific responsibilities.

Version 7

14.1 Segment the Network Based on Sensitivity

Segment the network based on the label or classification level of the information stored on the servers, locate all sensitive information on separated Virtual Local Area Networks (VLANs).

14.2 Enable Firewall Filtering Between VLANs

Enable firewall filtering between VLANs to ensure that only authorized systems are able to communicate with other systems necessary to fulfill their specific responsibilities.

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:

Version 7

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

14.8 Encrypt Sensitive Information at Rest
Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.

5.4.2 Consider external secret storage (Manual)

Profile Applicability:

- Level 2 - 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:

Version 7

14.8 Encrypt Sensitive Information at Rest
Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.

5.5 Extensible Admission Control

5.5.1 Configure Image Provenance using 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:

Version 6

18 Application Software Security
Application Software Security

5.7 General Policies

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

5.7.1 Create administrative boundaries between resources using namespaces (Manual)

Profile Applicability:

- Level 1 - 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 two initial namespaces:

1. `default` - The default namespace for objects with no other namespace
2. `kube-system` - The namespace for objects created by the Kubernetes system

References:

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

CIS Controls:

Version 6

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

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. Also, you need to enable all alpha features for this to work. There is no individual switch to turn on this feature.

Audit:

Review the pod definitions in your cluster. It should create a line as below:

```
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
```

Remediation:

Seccomp is an alpha feature currently. By default, all alpha features are disabled. So, you would need to enable alpha features in the apiserver by passing `"--feature-gates=AllAlpha=true"` argument.
Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--feature-gates=AllAlpha=true"`

```
KUBE_API_ARGS="--feature-gates=AllAlpha=true"
```

Based on your system, restart the `kube-apiserver` service. For example:

```
systemctl restart kube-apiserver.service
```

Use `annotations` to enable the `docker/default` seccomp profile in your pod definitions. An example is as below:

```
apiVersion: v1
kind: Pod
metadata:
  name: trustworthy-pod
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
spec:
  containers:
    - name: trustworthy-container
      image: sotrustworthy:latest
```

Default Value:

By default, seccomp profile is set to `unconfined` which means that no seccomp profiles are enabled.

References:

1. https://github.com/kubernetes/kubernetes/issues/39845
2. https://github.com/kubernetes/kubernetes/pull/21790
3. https://github.com/kubernetes/community/blob/master/contributors/design-proposals/seccomp.md#examples
4. https://docs.docker.com/engine/security/seccomp/

CIS Controls:

Version 6

5 Controlled Use of Administration Privileges
Controlled Use of Administration Privileges

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

Profile Applicability:

- Level 2 - 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:

Version 6

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

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

5.7.4 The default namespace should not be used (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 all
```

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

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Control Plane Components**		
1.1	**Master Node Configuration Files**		
1.1.1	Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.2	Ensure that the API server pod specification file ownership is set to root:root (Automated)	☐	☐
1.1.3	Ensure that the controller manager pod specification file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.4	Ensure that the controller manager pod specification file ownership is set to root:root (Automated)	☐	☐
1.1.5	Ensure that the scheduler pod specification file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.6	Ensure that the scheduler pod specification file ownership is set to root:root (Automated)	☐	☐
1.1.7	Ensure that the etcd pod specification file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.8	Ensure that the etcd pod specification file ownership is set to root:root (Automated)	☐	☐
1.1.9	Ensure that the Container Network Interface file permissions are set to 644 or more restrictive (Manual)	☐	☐
1.1.10	Ensure that the Container Network Interface file ownership is set to root:root (Manual)	☐	☐
1.1.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive (Automated)	☐	☐
1.1.12	Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)	☐	☐
1.1.13	Ensure that the admin.conf file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.14	Ensure that the admin.conf file ownership is set to root:root (Automated)	☐	☐
1.1.15	Ensure that the scheduler.conf file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.16	Ensure that the scheduler.conf file ownership is set to root:root (Automated)	☐	☐
1.1.17	Ensure that the controller-manager.conf file permissions are set to 644 or more restrictive (Automated)	☐	☐
1.1.18	Ensure that the controller-manager.conf file ownership is set to root:root (Automated)	☐	☐

1.1.19	Ensure that the 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 644 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 --basic-auth-file argument is not set (Automated)	☐	☐
1.2.3	Ensure that the --token-auth-file parameter is not set (Automated)	☐	☐
1.2.4	Ensure that the --kubelet-https argument is set to true (Automated)	☐	☐
1.2.5	Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Automated)	☐	☐
1.2.6	Ensure that the --kubelet-certificate-authority argument is set as appropriate (Automated)	☐	☐
1.2.7	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	☐	☐
1.2.8	Ensure that the --authorization-mode argument includes Node (Automated)	☐	☐
1.2.9	Ensure that the --authorization-mode argument includes RBAC (Automated)	☐	☐
1.2.10	Ensure that the admission control plugin EventRateLimit is set (Manual)	☐	☐
1.2.11	Ensure that the admission control plugin AlwaysAdmit is not set (Automated)	☐	☐
1.2.12	Ensure that the admission control plugin AlwaysPullImages is set (Manual)	☐	☐
1.2.13	Ensure that the admission control plugin SecurityContextDeny is set if PodSecurityPolicy is not used (Manual)	☐	☐
1.2.14	Ensure that the admission control plugin ServiceAccount is set (Automated)	☐	☐
1.2.15	Ensure that the admission control plugin NamespaceLifecycle is set (Automated)	☐	☐
1.2.16	Ensure that the admission control plugin PodSecurityPolicy is set (Automated)	☐	☐
1.2.17	Ensure that the admission control plugin NodeRestriction is set (Automated)	☐	☐
1.2.18	Ensure that the --insecure-bind-address argument is not set (Automated)	☐	☐

1.2.19	Ensure that the --insecure-port argument is set to 0 (Automated)	☐	☐
1.2.20	Ensure that the --secure-port argument is not set to 0 (Automated)	☐	☐
1.2.21	Ensure that the --profiling argument is set to false (Automated)	☐	☐
1.2.22	Ensure that the --audit-log-path argument is set (Automated)	☐	☐
1.2.23	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Automated)	☐	☐
1.2.24	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Automated)	☐	☐
1.2.25	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Automated)	☐	☐
1.2.26	Ensure that the --request-timeout argument is set as appropriate (Automated)	☐	☐
1.2.27	Ensure that the --service-account-lookup argument is set to true (Automated)	☐	☐
1.2.28	Ensure that the --service-account-key-file argument is set as appropriate (Automated)	☐	☐
1.2.29	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Automated)	☐	☐
1.2.30	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	☐	☐
1.2.31	Ensure that the --client-ca-file argument is set as appropriate (Automated)	☐	☐
1.2.32	Ensure that the --etcd-cafile argument is set as appropriate (Automated)	☐	☐
1.2.33	Ensure that the --encryption-provider-config argument is set as appropriate (Manual)	☐	☐
1.2.34	Ensure that encryption providers are appropriately configured (Manual)	☐	☐
1.2.35	Ensure that the API Server only makes use of Strong Cryptographic Ciphers (Manual)	☐	☐
1.3	**Controller Manager**		
1.3.1	Ensure that the --terminated-pod-gc-threshold argument is set as appropriate (Manual)	☐	☐
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)	☐	☐
2.6	Ensure that the --peer-auto-tls argument is not set to true (Automated)	☐	☐
2.7	Ensure that a unique Certificate Authority is used for etcd (Manual)	☐	☐
3	**Control Plane Configuration**		
3.1	**Authentication and Authorization**		
3.1.1	Client certificate authentication should not be used for users (Manual)	☐	☐
3.2	**Logging**		
3.2.1	Ensure that a minimal audit policy is created (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 644 or more restrictive (Automated)	☐	☐
4.1.2	Ensure that the kubelet service file ownership is set to root:root (Automated)	☐	☐
4.1.3	If proxy kubeconfig file exists ensure permissions are set to 644 or more restrictive (Manual)	☐	☐
4.1.4	If proxy kubeconfig file exists ensure ownership is set to root:root (Manual)	☐	☐

4.1.5	Ensure that the --kubeconfig kubelet.conf file permissions are set to 644 or more restrictive (Automated)	☐	☐
4.1.6	Ensure that the --kubeconfig kubelet.conf file ownership is set to root:root (Manual)	☐	☐
4.1.7	Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Manual)	☐	☐
4.1.8	Ensure that the client certificate authorities file ownership is set to root:root (Manual)	☐	☐
4.1.9	Ensure that the kubelet --config configuration file has permissions set to 644 or more restrictive (Automated)	☐	☐
4.1.10	Ensure that the kubelet --config configuration file ownership is set to root:root (Automated)	☐	☐
4.2	**Kubelet**		
4.2.1	Ensure that the --anonymous-auth argument is set to false (Automated)	☐	☐
4.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	☐	☐
4.2.3	Ensure that the --client-ca-file argument is set as appropriate (Automated)	☐	☐
4.2.4	Verify that the --read-only-port 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 --protect-kernel-defaults argument is set to true (Automated)	☐	☐
4.2.7	Ensure that the --make-iptables-util-chains argument is set to true (Automated)	☐	☐
4.2.8	Ensure that the --hostname-override argument is not set (Manual)	☐	☐
4.2.9	Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Manual)	☐	☐
4.2.10	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Manual)	☐	☐
4.2.11	Ensure that the --rotate-certificates argument is not set to false (Manual)	☐	☐
4.2.12	Verify that the RotateKubeletServerCertificate argument is set to true (Manual)	☐	☐
4.2.13	Ensure that the Kubelet only makes use of Strong Cryptographic Ciphers (Manual)	☐	☐
5	**Policies**		
5.1	**RBAC and Service Accounts**		
5.1.1	Ensure that the cluster-admin role is only used where required (Manual)	☐	☐
5.1.2	Minimize access to secrets (Manual)	☐	☐

5.1.3	Minimize wildcard use in Roles and ClusterRoles (Manual)	☐	☐
5.1.4	Minimize access to create pods (Manual)	☐	☐
5.1.5	Ensure that default service accounts are not actively used. (Manual)	☐	☐
5.1.6	Ensure that Service Account Tokens are only mounted where necessary (Manual)	☐	☐
5.2	**Pod Security Policies**		
5.2.1	Minimize the admission of privileged containers (Manual)	☐	☐
5.2.2	Minimize the admission of containers wishing to share the host process ID namespace (Manual)	☐	☐
5.2.3	Minimize the admission of containers wishing to share the host IPC namespace (Manual)	☐	☐
5.2.4	Minimize the admission of containers wishing to share the host network namespace (Manual)	☐	☐
5.2.5	Minimize the admission of containers with allowPrivilegeEscalation (Manual)	☐	☐
5.2.6	Minimize the admission of root containers (Manual)	☐	☐
5.2.7	Minimize the admission of containers with the NET_RAW capability (Manual)	☐	☐
5.2.8	Minimize the admission of containers with added capabilities (Manual)	☐	☐
5.2.9	Minimize the admission of containers with capabilities assigned (Manual)	☐	☐
5.3	**Network Policies and CNI**		
5.3.1	Ensure that the CNI in use supports Network Policies (Manual)	☐	☐
5.3.2	Ensure that all Namespaces have Network Policies defined (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)	☐	☐
5.7.3	Apply Security Context to Your Pods and Containers (Manual)	☐	☐
5.7.4	The default namespace should not be used (Manual)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
09/30/2020	1.6.0	Automated Audit Content added