Center for Internet Security®

CIS Kubernetes 1.6 Benchmark

v1.0.0 - 05-03-2017

Table of Contents

Overview

This document, CIS Kubernetes 1.6 Benchmark v1.0.0, provides prescriptive guidance for establishing a secure configuration posture for Kubernetes 1.6. To obtain the latest version of this guide, please visit. If you have questions, comments, or have identified ways to improve this guide, please write us at feedback@cisecurity.org.

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 1.6.

Consensus Guidance

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

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

Scoring Information

A scoring status indicates whether compliance with the given recommendation impacts the assessed target's benchmark score. The following scoring statuses are used in this benchmark:

Scored

Failure to comply with "Scored" recommendations will decrease the final benchmark score. Compliance with "Scored" recommendations will increase the final benchmark score.

Not Scored

Failure to comply with "Not Scored" recommendations will not decrease the final benchmark score. Compliance with "Not Scored" recommendations will not increase the final benchmark score.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

  Items in this profile intend to:

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

- **Level 2**

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

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

Acknowledgements

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

Authors
Pravin Goyal
Rory McCune
Jordan Liggit

Contributors
Puja Abbassi
Eric Chiang
Liron Levin
Jordan Rakoske

Recommendations

1 Master Node Security Configuration

This section consists of security recommendation for components on the master nodes.

1.1 API Server

This section contains recommendations for kube-apiserver configuration.

1.1.1 Ensure that the --allow-privileged argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Do not allow privileged containers.

Rationale:

The privileged container has all the system capabilities, and it also lifts all the limitations enforced by the device cgroup controller. In other words, the container can then do almost everything that the host can do. This flag exists to allow special use-cases, like running Docker within Docker and hence should be avoided for production workloads.

Audit:

Run the following command on the master node:

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

Verify that the `--allow-privileged` argument is set to `false`.

Remediation:

Edit the `/etc/kubernetes/config` file on the master node and set the `KUBE_ALLOW_PRIV` parameter to `"--allow-privileged=false"`:

```
KUBE_ALLOW_PRIV="--allow-privileged=false"
```

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

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

Impact:

You will not be able to run any privileged containers.

Note: A number of components used by Kubernetes clusters currently make use of privileged containers (e.g. Container Network Interface plugins). Care should be taken in ensuring that the use of such plugins is minimized and in particular any use of privileged containers outside of the kube-system namespace should be scrutinized. Where possible, review the rights required by such plugins to determine if a more fine grained permission set can be applied.

Default Value:

By default, privileged containers are not allowed.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/user-guide/security-context/

CIS Controls:

5 Controlled Use of Administration Privileges
Controlled Use of Administration Privileges

1.1.2 Ensure that the --anonymous-auth argument is set to false (Scored)

Profile Applicability:

- Level 1

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.

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--anonymous-auth=false"`:

```
KUBE_API_ARGS="--anonymous-auth=false"
```

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

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

Impact:

Anonymous requests will be rejected.

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:

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

1.1.3 Ensure that the --basic-auth-file argument is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not use basic authentication.

Rationale:

Basic authentication uses plaintext credentials for authentication. Currently, the basic authentication credentials last indefinitely, and the password cannot be changed without restarting API server. The basic authentication is currently supported for convenience. Hence, basic authentication should not 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 `/etc/kubernetes/apiserver` file on the master node and remove the `"--basic-auth-file=<filename>"` argument from the `KUBE_API_ARGS` parameter.

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

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

Impact:

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

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:

16 Account Monitoring and Control
Account Monitoring and Control

1.1.4 Ensure that the --insecure-allow-any-token argument is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not allow any insecure tokens

Rationale:

Accepting insecure tokens would allow any token without actually authenticating anything. User information is parsed from the token and connections are allowed.

Audit:

Run the following command on the master node:

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

Verify that the `--insecure-allow-any-token` argument does not exist.

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and remove the `--insecure-allow-any-token` argument from the `KUBE_API_ARGS` parameter.

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

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

Impact:

None

Default Value:

By default, insecure tokens are not allowed.

References:

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

CIS Controls:

16 Account Monitoring and Control
Account Monitoring and Control

1.1.5 Ensure that the --kubelet-https argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

Use https for kubelet connections.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and remove the `--kubelet-https` argument from the `KUBE_API_ARGS` parameter.

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

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

Impact:

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

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:

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

1.1.6 Ensure that the --insecure-bind-address argument is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not bind to non-loopback insecure addresses.

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 neither the insecure traffic is encrypted. Hence, you should not bind the apiserver to an insecure address.

Audit:

Run the following command on the master node:

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

Verify that the `--insecure-bind-address` argument does not exist or is set to 127.0.0.1.

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and remove the `--insecure-bind-address` argument from the `KUBE_API_ADDRESS` parameter.

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

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

Impact:

None

Default Value:

By default, insecure bind address is set to 127.0.0.1.

References:

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

CIS Controls:

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

1.1.7 Ensure that the --insecure-port argument is set to 0 (Scored)

Profile Applicability:

- Level 1

Description:

Do not bind to insecure port.

Rationale:

Setting up the apiserver to serve on an insecure port would allow unauthenticated and unencrypted access to your master node. It is assumed that firewall rules are set up such that this port is not reachable from outside of the cluster. But, as a defense in depth measure, you should not use an insecure port.

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 `/etc/kubernetes/apiserver` file on the master node and set `--insecure-port=0` in the `KUBE_API_PORT` parameter.

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

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

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

Default Value:

By default, the insecure port is set to 8080.

References:

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

CIS Controls:

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

1.1.8 Ensure that the --secure-port argument is not set to 0 (Scored)

Profile Applicability:

- Level 1

Description:

Do not disable the secure port.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and either remove the `--secure-port` argument from the `KUBE_API_ARGS` parameter or set it to a different desired port.

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

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

Impact:

You need to set the apiserver up with the right TLS certificates.

Default Value:

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

References:

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

CIS Controls:

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

1.1.9 Ensure that the --profiling argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Disable profiling, if not needed.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--profiling=false"`:

```
KUBE_API_ARGS="--profiling=false"
```

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

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

Impact:

Profiling information would not be available.

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:

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

1.1.10 Ensure that the --repair-malformed-updates argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Disable fixing of malformed updates.

Rationale:

The apiserver will potentially attempt to fix the update requests to pass the validation even if the requests are malformed. Malformed requests are one of the potential ways to interact with a service without legitimate information. Such requests could potentially be used to sabotage apiserver responses.

Audit:

Run the following command on the master node:

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

Verify that the `--repair-malformed-updates` argument is set to `false`.

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--repair-malformed-updates=false"`:

```
KUBE_API_ARGS="--repair-malformed-updates=false"
```

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

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

Impact:

Malformed requests from clients would be rejected.

Default Value:

By default, malformed updates are allowed.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://github.com/kubernetes/kubernetes/issues/15580

CIS Controls:

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

1.1.11 Ensure that the admission control policy is not set to AlwaysAdmit (Scored)

Profile Applicability:

- Level 1

Description:

Do not allow all requests.

Rationale:

Setting admission control policy to `AlwaysAdmit` allows all requests and do not filter any requests.

Audit:

Run the following command on the master node:

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

Verify that the `--admission-control` argument is set to a value that does not include `AlwaysAdmit`.

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_ADMISSION_CONTROL` parameter to a value that does not include `AlwaysAdmit`.

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

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

Impact:

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

Default Value:

By default, `AlwaysAdmit` is used if no `--admission-control` flag is provided.

References:

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

CIS Controls:

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

1.1.12 Ensure that the admission control policy is set to AlwaysPullImages (Scored)

Profile Applicability:

- Level 1

Description:

Always pull images.

Rationale:

Setting admission control policy to `AlwaysPullImages` forces every new pod to pull the required images every time. In a multitenant cluster users can be assured that their private images can only be used by those who have the credentials to pull them. Without this admisssion 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.

Audit:

Run the following command on the master node:

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

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

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_ADMISSION_CONTROL` parameter to `"--admission-control=...,AlwaysPullImages,..."`:

```
KUBE_ADMISSION_CONTROL="--admission-control=...,AlwaysPullImages,..."
```

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

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

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.

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:

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

1.1.13 Ensure that the admission control policy is set to DenyEscalatingExec (Scored)

Profile Applicability:

- Level 1

Description:

Deny execution of `exec` and `attach` commands in privileged pods.

Rationale:

Setting admission control policy to `DenyEscalatingExec` denies `exec` and `attach` commands to pods that run with escalated privileges that allow host access. This includes pods that run as privileged, have access to the host IPC namespace, and have access to the host PID namespace.

Audit:

Run the following command on the master node:

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

Verify that the `--admission-control` argument is set to a value that includes `DenyEscalatingExec`.

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_ADMISSION_CONTROL` parameter to `"--admission-control=...,DenyEscalatingExec,..."`:

```
KUBE_ADMISSION_CONTROL="--admission-control=...,DenyEscalatingExec,..."
```

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

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

Impact:

`exec` and `attach` commands will not work in privileged pods.

Default Value:

By default, `DenyEscalatingExec` is not set.

References:

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

CIS Controls:

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

1.1.14 Ensure that the admission control policy is set to SecurityContextDeny (Scored)

Profile Applicability:

- Level 1

Description:

Restrict pod level SecurityContext customization. Instead of using a customized SecurityContext for your pods, use a Pod Security Policy (PSP), which is a cluster-level resource that controls the actions that a pod can perform and what it has the ability to access.

Rationale:

Setting admission control policy to `SecurityContextDeny` denies the pod level SecurityContext customization. Any attempts to customize the SecurityContexts that are not explicitly defined in the Pod Security Policy (PSP) are blocked. This ensures that all the pods adhere to the PSP defined by your organization and you have a uniform pod level security posture.

Audit:

Run the following command on the master node:

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

Verify that the `--admission-control` argument is set to a value that includes `SecurityContextDeny`.

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_ADMISSION_CONTROL` parameter to `"--admission-control=...,SecurityContextDeny,..."`:

```
KUBE_ADMISSION_CONTROL="--admission-control=...,SecurityContextDeny,..."
```

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

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

Impact:

None

Default Value:

By default, `SecurityContextDeny` is 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:

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

1.1.15 Ensure that the admission control policy is set to NamespaceLifecycle (Scored)

Profile Applicability:

- Level 1

Description:

Reject creating objects in a namespace that is undergoing termination.

Rationale:

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

Audit:

Run the following command on the master node:

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

Verify that the `--admission-control` argument is set to a value that includes `NamespaceLifecycle`.

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_ADMISSION_CONTROL` parameter to `"--admission-control=NamespaceLifecycle,..."`:

```
KUBE_ADMISSION_CONTROL="--admission-control=NamespaceLifecycle,..."
```

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

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

Impact:

None

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:

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

1.1.16 Ensure that the --audit-log-path argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Enable auditing on kubernetes apiserver and set the desired audit log path as appropriate.

Rationale:

Auditing Kubernetes apiserver 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.

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--audit-log-path=<filename>"`:

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

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

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

Impact:

None

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:

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--audit-log-maxage=30"`:

```
KUBE_API_ARGS="--audit-log-maxage=30"
```

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

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

Impact:

None

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:

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

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

Profile Applicability:

- Level 1

Description:

Retain 10 or an appropriate number of old log files.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--audit-log-maxbackup=10"`:

```
KUBE_API_ARGS="--audit-log-maxbackup=10"
```

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

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

Impact:

None

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:

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

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

Profile Applicability:

- Level 1

Description:

Rotate log files on reaching 100 MB or as appropriate.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--audit-log-maxsize=100"`:

```
KUBE_API_ARGS="--audit-log-maxsize=100"
```

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

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

Impact:

None

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:

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

1.1.20 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)

Profile Applicability:

- Level 1

Description:

Do not always authorize all requests.

Rationale:

The apiserver, by default, allows all requests. You should restrict this behavior to only allow the authorization modes that you explicitly use in your environment. For example, if you don't use REST APIs in your environment, it is a good security best practice to switch off that capability.

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to values other than `--authorization-mode=AlwaysAllow`. One such example could be as below:

```
KUBE_API_ARGS="--authorization-mode=RBAC"
```

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

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

Impact:

Only authorized requests will be served.

Default Value:

By default, `AlwaysAllow` is enabled.

References:

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

CIS Controls:

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

1.1.21 Ensure that the --token-auth-file parameter is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not use token based authentication.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and remove the `"--token-auth-file=<filename>"` argument from the `KUBE_API_ARGS` parameter.

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

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

Impact:

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

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:

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

1.1.22 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Verify kubelet's certificate before establishing connection.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--kubelet-certificate-authority=<ca-string>"`:

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

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

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

Impact:

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

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:

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

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

Profile Applicability:

- Level 1

Description:

Enable certificate based kubelet authentication.

Rationale:

The apiserver, by default, does not authenticate kubelet's HTTPS endpoints. The requests from the apiserver are treated anonymously. You should setup certificate based kubelet authnetication to ensure that the apiserver authenticates the endpoints before submitting any requests.

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 setup the TLS connection between the apiserver and kubelets. Then, edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--kubelet-client-certificate=<path/to/client-certificate-file>"` and `"--kubelet-client-key=<path/to/client-key-file>"`:

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

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

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

Impact:

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

Default Value:

By default, certificate based kublete 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:

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

1.1.24 Ensure that the --service-account-lookup argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

Validate service account before validating token.

Rationale:

By default, the apiserver only verifies that the authentication token is valid. However, it 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.

Audit:

Run the following command on the master node:

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

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

Remediation:

Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--service-account-lookup=true"`:

```
KUBE_API_ARGS="--service-account-lookup=true"
```

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

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

Impact:

None

Default Value:

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

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:

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

1.1.25 Ensure that the admission control policy is set to PodSecurityPolicy (Scored)

Profile Applicability:

- Level 1

Description:

Reject creating pods that do not match Pod Security Policies.

Rationale:

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

Audit:

Run the following command on the master node:

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

Verify that the `--admission-control` 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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_ADMISSION_CONTROL` parameter to `"--admission-control=...,PodSecurityPolicy,..."`:

```
KUBE_ADMISSION_CONTROL="--admission-control=...,PodSecurityPolicy,..."
```

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

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

Impact:

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

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:

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

1.1.26 Ensure that the --service-account-key-file argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to `"--service-account-key-file=<filename>"`:

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

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

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

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.

Default Value:

By default, `--service-account-key-file` argument is not set, and the private key from the TLS serving certificate is used.

References:

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

CIS Controls:

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

1.1.27 Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to include `"--etcd-certfile=<path/to/client-certificate-file>"` and `"--etcd-keyfile=<path/to/client-key-file>"`:

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

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

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

Impact:

TLS and client certificate authentication must be configured for etcd.

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:

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

1.1.28 Ensure that the admission control policy is set to ServiceAccount (Scored)

Profile Applicability:

- Level 1

Description:

Automate service accounts management.

Rationale:

When you create a pod, if you do not specify a service account, it is automatically assigned the `default` service account in the same namespace. You should create your own service account and let the API server manage its security tokens.

Audit:

Run the following command on the master node:

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

Verify that the `--admission-control` argument is set to a value that includes `ServiceAccount`.

Remediation:

Follow the documentation and create `ServiceAccount` objects as per your environment. Then, edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_ADMISSION_CONTROL` parameter to `"--admission-control=...,ServiceAccount,..."`:

```
KUBE_ADMISSION_CONTROL="--admission-control=...,ServiceAccount,..."
```

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

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

Impact:

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

Default Value:

By default, `ServiceAccount` is not 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:

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

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

Profile Applicability:

- Level 1

Description:

Setup TLS connection on the API server.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to include `"--tls-cert-file=<path/to/tls-certificate-file>"` and `"--tls-private-key-file=<path/to/tls-key-file>"`:

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

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

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

Impact:

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

Default Value:

By default, `--tls-cert-file` and `--tls-private-key-file` arguments are not set. If HTTPS serving is enabled, and `--tls-cert-file` and `--tls-private-key-file` are not provided, a self-signed certificate and key are generated for the public address and saved to `/var/run/kubernetes`.

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:

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

1.1.30 Ensure that the --client-ca-file argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Setup TLS connection on the API server.

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to include `"--client-ca-file=<path/to/client-ca-file>"`:

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

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

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

Impact:

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

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:

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

1.1.31 Ensure that the --etcd-cafile argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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 `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to include `"--etcd-cafile=<path/to/ca-file>"`:

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

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

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

Impact:

TLS and client certificate authentication must be configured for etcd.

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:

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

1.2 Scheduler

This section contains recommendations for kube-scheduler configuration.

1.2.1 Ensure that the --profiling argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Disable profiling, if not needed.

Rationale:

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

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 `/etc/kubernetes/scheduler` file on the master node and set the `KUBE_SCHEDULER_ARGS` parameter to `"--profiling=false"`:

```
KUBE_SCHEDULER_ARGS="--profiling=false"
```

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

```
systemctl restart kube-scheduler.service
```

Impact:

Profiling information would not be available.

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:

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

1.3 Controller Manager

This section contains recommendations for kube-controller-manager configuration.

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

Profile Applicability:

- Level 1

Description:

Activate garbage collector on pod termination, as appropriate.

Rationale:

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

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 `/etc/kubernetes/controller-manager` file on the master node and set the `KUBE_CONTROLLER_MANAGER_ARGS` parameter to `"--terminated-pod-gc-threshold=<appropriate-number>"`:

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

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

```
systemctl restart kube-controller-manager.service
```

Impact:

None

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:

2 Inventory of Authorized and Unauthorized Software
Inventory of Authorized and Unauthorized Software

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

Profile Applicability:

- Level 1

Description:

Disable profiling, if not needed.

Rationale:

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

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 `/etc/kubernetes/controller-manager` file on the master node and set the `KUBE_CONTROLLER_MANAGER_ARGS` parameter to `"--profiling=false"`:

```
KUBE_CONTROLLER_MANAGER_ARGS="--profiling=false"
```

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

```
systemctl restart kube-controller-manager.service
```

Impact:

Profiling information would not be available.

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:

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

1.3.3 Ensure that the --insecure-experimental-approve-all-kubelet-csrs-for-group argument is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not accept all certificates.

Rationale:

Setting the `--insecure-experimental-approve-all-kubelet-csrs-for-group` flag circumvents the desired "approval" process. All the certificates are auto-approved without checking their integrity. This flag is meant to be used for development and testing purposes only and hence should not be used in the production.

Audit:

Run the following command on the master node:

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

Verify that the `--insecure-experimental-approve-all-kubelet-csrs-for-group` argument is not set.

Remediation:

Edit the `/etc/kubernetes/controller-manager` file on the master node and remove the `--insecure-experimental-approve-all-kubelet-csrs-for-group` argument from the `KUBE_CONTROLLER_MANAGER_ARGS` parameter.

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

```
systemctl restart kube-controller-manager.service
```

Impact:

Invalid certificates will be rejected.

Default Value:

By default, `--insecure-experimental-approve-all-kubelet-csrs-for-group` is not set.

References:

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

CIS Controls:

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

1.3.4 Ensure that the --use-service-account-credentials argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

Use individual service account credentials for each controller.

Rationale:

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

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 `/etc/kubernetes/controller-manager` file on the master node and set the `KUBE_CONTROLLER_MANAGER_ARGS` parameter to `--use-service-account-credentials=true`:

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

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

```
systemctl restart kube-controller-manager.service
```

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.

Default Value:

By default, `--use-service-account-credentials` is not set.

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:

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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 `/etc/kubernetes/controller-manager` file on the master node and set the `KUBE_CONTROLLER_MANAGER_ARGS` parameter to `--service-account-private-key-file=<filename>`:

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

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

```
systemctl restart kube-controller-manager.service
```

Impact:

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

Default Value:

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

References:

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

CIS Controls:

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

1.3.6 Ensure that the --root-ca-file argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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

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 `/etc/kubernetes/controller-manager` file on the master node and set the `KUBE_CONTROLLER_MANAGER_ARGS` parameter to include `--root-ca-file=<file>`:

```
KUBE_CONTROLLER_MANAGER_ARGS="--root-ca-file=<file>"
```

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

```
systemctl restart kube-controller-manager.service
```

Impact:

You need to setup and maintain root certificate authority 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:

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

1.4 Configuration Files

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

1.4.1 Ensure that the apiserver file permissions are set to 644 or more restrictive (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `apiserver` 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.

Audit:

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

```
stat -c %a /etc/kubernetes/apiserver
```

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/apiserver
```

Impact:

None

Default Value:

By default, `apiserver` file has permissions of `644`.

References:

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

CIS Controls:

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

1.4.2 Ensure that the apiserver file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `apiserver` 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`.

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/apiserver
```

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/apiserver
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

14 Controlled Access Based on the Need to Know

Controlled Access Based on the Need to Know

1.4.3 Ensure that the config file permissions are set to 644 or more restrictive (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `config` file controls various parameters that set the behavior of various components of 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.

Audit:

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

```
stat -c %a /etc/kubernetes/config
```

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/config
```

Impact:

None

Default Value:

By default, `config` file has permissions of `644`.

References:

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

CIS Controls:

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

1.4.4 Ensure that the config file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `config` 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`.

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/config
```

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/config
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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

1.4.5 Ensure that the scheduler file permissions are set to 644 or more restrictive (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `scheduler` file controls various parameters that set the behavior of the `kube-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.

Audit:

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

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

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
```

Impact:

None

Default Value:

By default, `scheduler` file has permissions of `644`.

References:

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

CIS Controls:

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

1.4.6 Ensure that the scheduler file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `scheduler` 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`.

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
```

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
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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

1.4.7 Ensure that the etcd.conf file permissions are set to 644 or more restrictive (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `etcd.conf` file 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.

Audit:

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

```
stat -c %a /etc/etcd/etcd.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/etcd/etcd.conf
```

Impact:

None

Default Value:

By default, `etcd.conf` file has permissions of `644`.

References:

1. https://coreos.com/etcd

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

CIS Controls:

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

1.4.8 Ensure that the etcd.conf file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `etcd.conf` file 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`.

Audit:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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

1.4.9 Ensure that the flanneld file permissions are set to 644 or more restrictive (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `flanneld` file controls various parameters that set the behavior of the `flanneld` service in the master node. Flannel is one of the various options for a simple overlay network. 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.

Audit:

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

```
stat -c %a /etc/sysconfig/flanneld
```

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

Note: Flannel is an optional component of Kubernetes. If you are not using Flannel then this requirement is not applicable. If you are using any other option for configuring your networking, please extend this recommendation to cover important configuration files as appropriate.

Remediation:

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

```
chmod 644 /etc/sysconfig/flanneld
```

Impact:

None

Default Value:

Note: Flannel is an optional component of Kubernetes and there are other alternatives that might be used in its place. Please checkout the Kubernetes documentation for other options.

If you are using Flannel for setting up your networking then, by default, `flanneld` file has permissions of `644`.

References:

1. https://coreos.com/flannel/docs/latest/
2. https://kubernetes.io/docs/concepts/cluster-administration/networking/#flannel

CIS Controls:

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

1.4.10 Ensure that the flanneld file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `flanneld` file controls various parameters that set the behavior of the `flanneld` service in the master node. Flannel is one of the various options for a simple overlay network. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Audit:

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

```
stat -c %U:%G /etc/sysconfig/flanneld
```

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

Note: Flannel is an optional component of Kubernetes. If you are not using Flannel then this requirement is not applicable. If you are using any other option for configuring your networking, please extend this recommendation to cover important configuration files as appropriate.

Remediation:

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

```
chown root:root /etc/sysconfig/flanneld
```

Impact:

None

Default Value:

Note: Flannel is an optional component of Kubernetes and there are other alternatives that might be used in its place. Please checkout the Kubernetes documentation for other options.

If you are using Flannel for setting up your networking then, by default, `flanneld` file ownership is set to `root:root`.

References:

1. https://coreos.com/flannel/docs/latest/
2. https://kubernetes.io/docs/concepts/cluster-administration/networking/#flannel

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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/default.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.etcd
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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

1.4.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Scored)

Profile Applicability:

- Level 1

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`.

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/default.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.etcd
```

Impact:

None

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:

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

1.5 etcd

This section covers recommendations for etcd configuration on the master nodes.

1.5.1 Ensure that the --cert-file and --key-file arguments are set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Configure TLS encryption for the etcd service.

Rationale:

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

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.

Impact:

Client connections only over TLS would be served.

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:

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

1.5.2 Ensure that the --client-cert-auth argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

Enable client authentication on etcd service.

Rationale:

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

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 envrironment file (for example, `/etc/etcd/etcd.conf`) on the etcd server node and set the `ETCD_CLIENT_CERT_AUTH` parameter to `"true"`:

```
ETCD_CLIENT_CERT_AUTH="true"
```

Edit the etcd startup file (for example, `/etc/systemd/system/multi-user.target.wants/etcd.service`) and configure the startup parameter for `--client-cert-auth` and set it to `\"${ETCD_CLIENT_CERT_AUTH}\"`:

```
ExecStart=/bin/bash -c "GOMAXPROCS=$(nproc) /usr/bin/etcd --
name=\"${ETCD_NAME}\" --data-dir=\"${ETCD_DATA_DIR}\" --listen-client-
urls=\"${ETCD_LISTEN_CLIENT_URLS}\" --client-cert-
auth=\"${ETCD_CLIENT_CERT_AUTH}\""
```

Based on your system, reload the daemon and restart the `etcd` service. For example,

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

Impact:

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

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:

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

1.5.3 Ensure that the --auto-tls argument is not set to true (Scored)

Profile Applicability:

- Level 1

Description:

Do not use self-signed certificates for TLS.

Rationale:

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

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 environment file (for example, `/etc/etcd/etcd.conf`) on the etcd server node and comment out the `ETCD_AUTO_TLS` parameter.

```
#ETCD_AUTO_TLS="true"
```

Edit the etcd startup file (for example, `/etc/systemd/system/multi-user.target.wants/etcd.service`) and remove the startup parameter for `--auto-tls`. Based on your system, reload the daemon and restart the `etcd` service. For example,

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

Impact:

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

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:

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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.

Impact:

etcd cluster peers would need to set up TLS for their communication.

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:

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

1.5.5 Ensure that the --peer-client-cert-auth argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

etcd should be configured for peer authentication.

Rationale:

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

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 environment file (for example, `/etc/etcd/etcd.conf`) on the etcd server node and set the `ETCD_PEER_CLIENT_CERT_AUTH` parameter to `"true"`:

```
ETCD_PEER_CLIENT_CERT_AUTH="true"
```

Edit the etcd startup file (for example, `/etc/systemd/system/multi-user.target.wants/etcd.service`) and configure the startup parameter for `--peer-client-cert-auth` and set it to `\"${ETCD_PEER_CLIENT_CERT_AUTH}\"`:

```
ExecStart=/bin/bash -c "GOMAXPROCS=$(nproc) /usr/bin/etcd --
name=\"${ETCD_NAME}\" --data-dir=\"${ETCD_DATA_DIR}\" --listen-client-
urls=\"${ETCD_LISTEN_CLIENT_URLS}\" --peer-client-cert-
auth=\"${ETCD_PEER_CLIENT_CERT_AUTH}\""
```

Based on your system, reload the daemon and restart the etcd service. For example,

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

Impact:

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

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:

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

1.5.6 Ensure that the --peer-auto-tls argument is not set to true (Scored)

Profile Applicability:

- Level 1

Description:

Do not use automatically generated self-signed certificates for TLS connections between peers.

Rationale:

etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. These objects are sensitive in nature and should be accessible only by authenticated etcd peers in the etcd cluster. Hence, do not use self-signed certificates for authentication.

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 environment file (for example, `/etc/etcd/etcd.conf`) on the etcd server node and comment out the `ETCD_PEER_AUTO_TLS` parameter:

```
#ETCD_PEER_AUTO_TLS="true"
```

Edit the etcd startup file (for example, `/etc/systemd/system/multi-user.target.wants/etcd.service`) and remove the startup parameter for `--peer-auto-tls`. Based on your system, reload the daemon and restart the etcd service. For example,

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

Impact:

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

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:

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

1.5.7 Ensure that the --wal-dir argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Store etcd logs separately from etcd data.

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 mixed with log data. Keeping the log data separate from the etcd data also ensures that those two types of data could individually be safeguarded. Also, you could use a centralized and remote log directory for persistent logging. Additionally, this separation also helps to avoid IO competition between logging and other IO operations.

Audit:

Run the following command on the etcd server node:

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

Verify that `--wal-dir` argument exists, and it is set as appropriate. At the minimum, it should not be set to the same directory as set for `--data-dir` argument.

Remediation:

Edit the etcd environment file (for example, `/etc/etcd/etcd.conf`) on the etcd server node and set the `ETCD_WAL_DIR` parameter as appropriate:

```
ETCD_WAL_DIR="<dir-name>"
```

Edit the etcd startup file (for example, `/etc/systemd/system/multi-user.target.wants/etcd.service`) and configure the startup parameter for `--wal-dir` and set it to `\"${ETCD_WAL_DIR}\"`:

```
ExecStart=/bin/bash -c "GOMAXPROCS=$(nproc) /usr/bin/etcd --
name=\"${ETCD_NAME}\" --data-dir=\"${ETCD_DATA_DIR}\" --listen-client-
urls=\"${ETCD_LISTEN_CLIENT_URLS}\" --wal-dir=\"${ETCD_WAL_DIR}\""
```

Based on your system, reload the daemon and restart the etcd service. For example,

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

Impact:

None

Default Value:

By default, `--wal-dir` argument is not set.

References:

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

CIS Controls:

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

1.5.8 Ensure that the --max-wals argument is set to 0 (Scored)

Profile Applicability:

- Level 1

Description:

Do not auto rotate logs.

Rationale:

etcd is a highly-available key value store used by Kubernetes deployments for persistent storage of all of its REST API objects. You should avoid automatic log rotation and instead safeguard the logs in a centralized repository or through a separate log management system.

Audit:

Run the following command on the etcd server node:

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

Verify that `--max-wals` argument exists and it is set to `0`.

Remediation:

Edit the etcd environment file (for example, `/etc/etcd/etcd.conf`) on the etcd server node and set the `ETCD_MAX_WALS` parameter to `0`:

```
ETCD_MAX_WALS="0"
```

Edit the etcd startup file (for example, `/etc/systemd/system/multi-user.target.wants/etcd.service`) and configure the startup parameter for `--max-wals` and set it to `\"${ETCD_MAX_WALS}\"`:

```
ExecStart=/bin/bash -c "GOMAXPROCS=$(nproc) /usr/bin/etcd --
name=\"${ETCD_NAME}\" --data-dir=\"${ETCD_DATA_DIR}\" --listen-client-
urls=\"${ETCD_LISTEN_CLIENT_URLS}\" --max-walsr=\"${ETCD_MAX_WALS}\""
```

Based on your system, reload the daemon and restart the etcd service. For example,

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

Impact:

You will have to manage log rotation and archiving.

Default Value:

By default, `--max-wals` argument is set to `5`.

References:

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

CIS Controls:

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

1.5.9 Ensure that a unique Certificate Authority is used for etcd (Not Scored)

Profile Applicability:

- Level 2

Description:

Use a different certificate authority for etcd from the one used for Kubernetes.

Rationale:

etcd is a highly available key-value store used by Kubernetes deployments for persistent storage of all of its REST API objects. Its access should be restricted to specifically designated clients and peers only.

Authentication to etcd is based on whether the certificate presented was issued by a trusted certificate authority. There is no checking of certificate attributes such as common name or subject alternative name. As such, if any attackers were able to gain access to any certificate issued by the trusted certificate authority, they would be able to gain full access to the etcd database.

Audit:

Review the CA used by the etcd environment and ensure that it does not match the CA certificate used by Kubernetes.

Run the following command on the etcd server node:

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

Review the file referenced by the `--trusted-ca-file` argument and ensure that the referenced CA is not the same one as is used for management of the overall Kubernetes cluster.

Remediation:

Follow the etcd documentation and create a dedicated certificate authority setup for the etcd service.

Impact:

Additional management of the certificates and keys for the dedicated certificate authority will be required.

Default Value:

NA

References:

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

CIS Controls:

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

1.6 General Security Primitives

This section contains general security features and controls provided by Kubernetes. These features can be used in various ways to tighten the security in Kubernetes environment. Due to varied nature of these configurations, only a few suggested approaches for configuring such controls are provided. The actual settings are site-specific in nature and are not scorable without manual intervention.

1.6.1 Ensure that the cluster-admin role is only used where required (Not Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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:

Remove any unneeded `clusterrolebindings`:

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

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.

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:

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.

1.6.2 Create Pod Security Policies for your cluster (Not Scored)

Profile Applicability:

- Level 1

Description:

Create and enforce Pod Security Policies for your cluster.

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.

Audit:

Run the below command and review the Pod Security Policies enforced on the cluster.

```
kubectl get psp
```

Ensure that these policies are configured as per your security requirements.

Remediation:

Follow the documentation and create and enforce Pod Security Policies for your cluster. Additionally, you could refer the "CIS Security Benchmark for Docker" and follow the suggested Pod Security Policies for your environment.

Impact:

Pods must align with the Pod Security Policies enforced on the cluster.

Default Value:

By default, Pod Security Policies are not created.

References:

1. https://kubernetes.io/docs/concepts/policy/pod-security-policy/
2. https://benchmarks.cisecurity.org/downloads/browse/index.cfm?category=benchmarks.servers.virtualization.docker

CIS Controls:

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

1.6.3 Create administrative boundaries between resources using namespaces (Not Scored)

Profile Applicability:

- Level 1

Description:

Use namespaces to isolate your Kubernetes objects.

Rationale:

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

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.

Impact:

You need to switch between namespaces for administration.

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:

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

1.6.4 Create network segmentation using Network Policies (Not Scored)

Profile Applicability:

- Level 2

Description:

Use network policies to isolate 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. `NetworkPolicy` resources use labels to select pods and define whitelist rules which allow traffic to the selected pods in addition to what is allowed by the isolation policy for a given namespace.

Audit:

Run the below command and review the `NetworkPolicy` objects created in the cluster.

```
kubectl get pods --namespace=kube-system
```

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

Remediation:

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

Impact:

You need a networking solution which supports `NetworkPolicy` - simply creating the resource without a controller to implement it will have no effect.

Default Value:

By default, network policies are not created.

References:

1. https://kubernetes.io/docs/concepts/services-networking/networkpolicies/

2. http://blog.kubernetes.io/2016/08/security-best-practices-kubernetes-deployment.html
3. https://kubernetes.io/docs/tasks/configure-pod-container/declare-network-policy/

CIS Controls:

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

Limitation and Control of Network Ports, Protocols, and Services

1.6.5 Avoid using Kubernetes Secrets (Not Scored)

Profile Applicability:

- Level 2

Description:

Avoid using Kubernetes `secret`.

Rationale:

Kubernetes objects of type `secret` are intended to hold sensitive information, such as passwords, OAuth tokens, and ssh keys. Its current implementation is very basic. It has plenty of risks as highlighted in the reference links including storing secrets as plaintext. Avoid using Kubernetes secrets until you have devised a mechanism to protect them using your own means.

Audit:

Run the below command and review if there are any `secret` objects created in the cluster.

```
kubectl get secrets
```

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

Remediation:

Use other mechanisms such as vaults to manage your cluster secrets.

Impact:

You need to use other mechanisms for managing secrets in your cluster.

Default Value:

By default, Kubernetes automatically creates secrets which contain credentials for accessing the API and it automatically modifies your pods to use this type of secret. Please note that those default token secrets are automatically created and deleting them won't be of any use, because Kubernetes will just recreate them.

References:

1. https://kubernetes.io/docs/concepts/configuration/secret/#risks
2. https://github.com/kubernetes/kubernetes/issues/10439

3. https://github.com/kubernetes/community/blob/master/contributors/design-proposals/secrets.md

CIS Controls:

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

1.6.6 Ensure that the seccomp profile is set to docker/default in your pod definitions (Not Scored)

Profile Applicability:

- Level 2

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.

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
```

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.

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:

18 Application Software Security
Application Software Security

1.6.7 Apply Security Context to Your Pods and Containers (Not Scored)

Profile Applicability:

- Level 2

Description:

Apply Security Context to Your Pods and Containers

Rationale:

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

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.

Impact:

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

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:

18 Application Software Security
Application Software Security

1.6.8 Configure Image Provenance using ImagePolicyWebhook admission controller (Not Scored)

Profile Applicability:

- Level 2

Description:

Configure Image Provenance for your deployment.

Rationale:

Kubernetes supports plugging in provenance rules to accept or reject the images in your deployments. You could configure such rules to ensure that only approved images are deployed in the cluster.

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.

Impact:

You need to regularly maintain your provenance configuration based on container image updates.

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:

18 Application Software Security
Application Software Security

2 Worker Node Security Configuration

This section consists of security recommendation for components on the worker nodes.

2.1 Kubelet

This section contains recommendations for kubelet configuration.

2.1.1 Ensure that the --allow-privileged argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Do not allow privileged containers.

Rationale:

The privileged container has all the system capabilities, and it also lifts all the limitations enforced by the device cgroup controller. In other words, the container can then do almost everything that the host can do. This flag exists to allow special use-cases, like running Docker within Docker and hence should be avoided for production workloads.

Audit:

Run the following command on each node:

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

Verify that the `--allow-privileged` argument is set to `false`.

Remediation:

Edit the `/etc/kubernetes/config` file on each node and set the `KUBE_ALLOW_PRIV` parameter to `"--allow-privileged=false"`:

```
KUBE_ALLOW_PRIV="--allow-privileged=false"
```

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

```
systemctl restart kubelet.service
```

Impact:

You will not be able to run any privileged containers.

Note: A number of components used by Kubernetes clusters currently make use of privileged containers (e.g. Container Network Interface plugins). Care should be taken in ensuring that the use of such plugins is minimized and in particular any use of privileged containers outside of the kube-system namespace should be scrutinized. Where possible, review the rights required by such plugins to determine if a more fine grained permission set can be applied.

Default Value:

By default, privileged containers are not allowed.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/user-guide/security-context/

CIS Controls:

5 Controlled Use of Administration Privileges
Controlled Use of Administration Privileges

2.1.2 Ensure that the --anonymous-auth argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Disable anonymous requests to the Kubelet server.

Rationale:

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

Audit:

Run the following command on each node:

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

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

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--anonymous-auth=false"`:

```
KUBELET_ARGS="--anonymous-auth=false"
```

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

```
systemctl restart kubelet.service
```

Impact:

Anonymous requests will be rejected.

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:

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

2.1.3 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)

Profile Applicability:

- Level 1

Description:

Do not allow all requests. Enable explicit authorization.

Rationale:

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

Audit:

Run the following command on each node:

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

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

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--authorization-mode=Webhook"`:

```
KUBELET_ARGS="--authorization-mode=Webhook"
```

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

```
systemctl restart kubelet.service
```

Impact:

Unauthorized requests will be denied.

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:

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

2.1.4 Ensure that the --client-ca-file argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Enable Kubelet authentication using certificates.

Rationale:

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

Audit:

Run the following command on each node:

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

Verify that the `--client-ca-file` 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 `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--client-ca-file=<path/to/client-ca-file>"`:

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

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

```
systemctl restart kubelet.service
```

Impact:

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

Default Value:

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

References:

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

CIS Controls:

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

2.1.5 Ensure that the --read-only-port argument is set to 0 (Scored)

Profile Applicability:

- Level 1

Description:

Disable the read-only port.

Rationale:

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

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`.

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--read-only-port=0"`

```
KUBELET_ARGS="--read-only-port=0"
```

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

```
systemctl restart kubelet.service
```

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.

Default Value:

By default, `--read-only-port` is set to `10255/TCP`.

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Do not disable timeouts on streaming connections.

Rationale:

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

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

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`.

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--streaming-connection-idle-timeout=<appropriate-timeout-value>"`

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

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

```
systemctl restart kubelet.service
```

Impact:

Long-lived connections could be interrupted.

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:

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

2.1.7 Ensure that the --protect-kernel-defaults argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

Run the following command on each node:

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

Verify that the `--protect-kernel-defaults` argument is set to `true`.

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--protect-kernel-defaults=true"`

```
KUBELET_ARGS="--protect-kernel-defaults=true"
```

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

```
systemctl restart kubelet.service
```

Impact:

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

Default Value:

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

References:

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

CIS Controls:

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

2.1.8 Ensure that the --make-iptables-util-chains argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

Allow Kubelet to manage iptables.

Rationale:

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

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`.

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and remove the `--make-iptables-util-chains` argument from the `KUBELET_ARGS` parameter. Based on your system, restart the `kubelet` service. For example:

```
systemctl restart kubelet.service
```

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.

Default Value:

By default, `--make-iptables-util-chains` argument is set to `true`.

References:

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

CIS Controls:

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

2.1.9 Ensure that the --keep-terminated-pod-volumes argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Unmount volumes from the nodes on pod termination.

Rationale:

On pod termination, you should unmount the volumes. Those volumes might have sensitive data that might be exposed if kept mounted on the node without any use. Additionally, such mounted volumes could be modified and later could be mounted on pods. Also, if you retain all mounted volumes for a long time, it might exhaust system resources and you might not be able to mount any more volumes on new pods.

Audit:

Run the following command on each node:

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

Verify that `--keep-terminated-pod-volumes` argument exists and is set to `false`.

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--keep-terminated-pod-volumes=false"`:

```
KUBELET_ARGS="--keep-terminated-pod-volumes=false"
```

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

```
systemctl restart kubelet.service
```

Impact:

Volumes will not be available for debugging.

Default Value:

By default, `--keep-terminated-pod-volumes` argument is set to `true`.

References:

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

CIS Controls:

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

2.1.10 Ensure that the --hostname-override argument is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not override node hostnames.

Rationale:

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

Audit:

Run the following command on each node:

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

Verify that `--hostname-override` argument does not exist.

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_HOSTNAME` parameter to `""`:

```
KUBELET_HOSTNAME=""
```

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

```
systemctl restart kubelet.service
```

Impact:

Node hostnames should have resolvable FQDNs.

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:

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

2.1.11 Ensure that the --event-qps argument is set to 0 (Scored)

Profile Applicability:

- Level 1

Description:

Do not limit event creation.

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.

Audit:

Run the following command on each node:

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

Verify that `--event-qps` argument exists and is set to `0`.

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--event-qps=0"`:

```
KUBELET_ARGS="--event-qps=0"
```

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

```
systemctl restart kubelet.service
```

Impact:

You might need to scale up your event storage and processing capabilitles.

Default Value:

By default, `--event-qps` argument is set to `5`.

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Setup TLS connection on the Kubelets.

Rationale:

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

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.

Remediation:

Follow the Kubernetes documentation and set up the TLS connection on the Kubelet. Then, edit the `/etc/kubernetes/kubelet` file on the master node and set the `KUBELET_ARGS` parameter to include `"--tls-cert-file=<path/to/tls-certificate-file>"` and `"--tls-private-key-file=<path/to/tls-key-file>"`:

```
KUBELET_ARGS="--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 restart kubelet.service
```

Impact:

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

Default Value:

By default, `--tls-cert-file` and `--tls-private-key-file` arguments are not set. If `--tls-cert-file` and `--tls-private-key-file` are not provided, a self-signed certificate and key are generated for the public address and saved to the directory passed to `--cert-dir`.

References:

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

CIS Controls:

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

2.1.13 Ensure that the --cadvisor-port argument is set to 0 (Scored)

Profile Applicability:

- Level 1

Description:

Disable cAdvisor.

Rationale:

cAdvisor provides potentially sensitive data and there's currently no way to block access to it using anything other than iptables. It does not require authentication/authorization to connect to the cAdvisor port. Hence, you should disable the port.

Audit:

Run the following command on each node:

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

Verify that `--cadvisor-port` argument exists and is set to `0`.

Remediation:

Edit the `/etc/kubernetes/kubelet` file on each node and set the `KUBELET_ARGS` parameter to `"--cadvisor-port=0"`:

```
KUBELET_ARGS="--cadvisor-port=0"
```

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

```
systemctl restart kubelet.service
```

Impact:

cAdvisor will not be available directly. You need to work with `/metrics` endpoint on the API server.

Default Value:

By default, `--cadvisor-port` argument is set to `4194`.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://github.com/kubernetes/kubernetes/issues/11710

3. https://github.com/kubernetes/kubernetes/issues/32638
4. https://raesene.github.io/blog/2016/10/14/Kubernetes-Attack-Surface-cAdvisor/

CIS Controls:

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

2.2 Configuration Files

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

2.2.1 Ensure that the config file permissions are set to 644 or more restrictive (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `config` file controls various parameters that set the behavior of various components 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.

Audit:

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

```
stat -c %a /etc/kubernetes/config
```

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/config
```

Impact:

None

Default Value:

By default, `config` file has permissions of `644`.

References:

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

CIS Controls:

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

2.2.2 Ensure that the config file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `config` file controls various parameters that set the behavior of various components 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`.

Audit:

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

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

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/config
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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

2.2.3 Ensure that the kubelet file permissions are set to 644 or more restrictive (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `kubelet` 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.

Audit:

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

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

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
```

Impact:

None

Default Value:

By default, `kubelet` file has permissions of `644`.

References:

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

CIS Controls:

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

2.2.4 Ensure that the kubelet file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `kubelet` 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`.

Audit:

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

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

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
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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

2.2.5 Ensure that the proxy file permissions are set to 644 or more restrictive (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `proxy` file controls various parameters that set the behavior 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.

Audit:

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

```
stat -c %a /etc/kubernetes/proxy
```

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/proxy
```

Impact:

None

Default Value:

By default, `proxy` file has permissions of `644`.

References:

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

CIS Controls:

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

2.2.6 Ensure that the proxy file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `proxy` file controls various parameters that set the behavior of 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`.

Audit:

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

```
stat -c %U:%G /etc/kubernetes/proxy
```

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/proxy
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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

3 Federated Deployments

This section contains recommendations for federated deployments.

3.1 Federation API Server

This section contains recommendations for federation-apiserver configuration.

3.1.1 Ensure that the --anonymous-auth argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

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

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `--anonymous-auth=false`.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

Anonymous requests will be rejected.

Default Value:

By default, anonymous access is enabled.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.2 Ensure that the --basic-auth-file argument is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not use basic authentication.

Rationale:

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

Audit:

Run the following command:

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

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

Remediation:

Follow the documentation and configure alternate mechanisms for authentication. Then, edit the deployment specs and remove `"--basic-auth-file=<filename>"`.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

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

Default Value:

By default, basic authentication is not set.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml

3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

16 Account Monitoring and Control
Account Monitoring and Control

3.1.3 Ensure that the --insecure-allow-any-token argument is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not allow any insecure tokens.

Rationale:

Accepting insecure tokens would allow any token without actually authenticating anything. User information is parsed from the token and connections are allowed.

Audit:

Run the following command:

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

Verify that the `--insecure-allow-any-token` argument does not exist.

Remediation:

Edit the deployment specs and remove `--insecure-allow-any-token`.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

By default, insecure tokens are not allowed.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

16 Account Monitoring and Control
Account Monitoring and Control

3.1.4 Ensure that the --insecure-bind-address argument is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not bind to insecure addresses.

Rationale:

If you bind the federation apiserver to an insecure address, basically anyone who could connect to it over the insecure port, would have unauthenticated and unencrypted access to it. The federation apiserver doesn't do any authentication checking for insecure binds and neither the insecure traffic is encrypted. Hence, you should not bind the federation apiserver to an insecure address.

Audit:

Run the following command:

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

Verify that the `--insecure-bind-address` argument does not exist or is set to 127.0.0.1.

Remediation:

Edit the deployment specs and remove `--insecure-bind-address`.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

By default, insecure bind address is set to 127.0.0.1.

References:

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

2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.5 Ensure that the --insecure-port argument is set to 0 (Scored)

Profile Applicability:

- Level 1

Description:

Do not bind to insecure port.

Rationale:

Setting up the federation apiserver to serve on an insecure port would allow unauthenticated and unencrypted access to it. It is assumed that firewall rules are set up such that this port is not reachable from outside of the cluster. But, as a defense in depth measure, you should not use an insecure port.

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `--insecure-port=0`.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

By default, the insecure port is set to 8080.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.6 Ensure that the --secure-port argument is not set to 0 (Scored)

Profile Applicability:

- Level 1

Description:

Do not disable the secure port.

Rationale:

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

Audit:

Run the following command:

```
ps -ef | grep federation-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 deployment specs and set the `--secure-port` argument to the desired port.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

You need to set the federation apiserver up with the right TLS certificates.

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.7 Ensure that the --profiling argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Disable profiling, if not needed.

Rationale:

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

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `"--profiling=false"`:

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

Profiling information would not be available.

Default Value:

By default, profiling is enabled.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.8 Ensure that the admission control policy is not set to AlwaysAdmit (Scored)

Profile Applicability:

- Level 1

Description:

Do not allow all requests.

Rationale:

Setting admission control policy to `AlwaysAdmit` allows all requests and do not filter any requests.

Audit:

Run the following command:

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

Verify that the `--admission-control` argument is set to a value that does not include `AlwaysAdmit`.

Remediation:

Edit the deployment specs and set `--admission-control` argument to a value that does not include `AlwaysAdmit`.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

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

Default Value:

By default, `AlwaysAdmit` is used if no `--admission-control` flag is provided.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml

3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.9 Ensure that the admission control policy is set to NamespaceLifecycle (Scored)

Profile Applicability:

- Level 1

Description:

Reject creating objects in a namespace that is undergoing termination.

Rationale:

Setting admission control policy to `NamespaceLifecycle` ensures that the 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.

Audit:

Run the following command:

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

Verify that the `--admission-control` argument is set to a value that includes `NamespaceLifecycle`.

Remediation:

Edit the deployment specs and set `--admission-control` argument to a value that includes `NamespaceLifecycle`.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

By default, `NamespaceLifecycle` is set.

References:

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

2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.10 Ensure that the --audit-log-path argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Enable auditing on kubernetes federation apiserver and set the desired audit log path as appropriate.

Rationale:

Auditing Kubernetes federation apiserver 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.

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `--audit-log-path` argument as appropriate.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

By default, auditing is not enabled.

References:

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

2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `--audit-log-maxage` to `30` or as appropriate.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

By default, auditing is not enabled.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Retain 10 or an appropriate number of old log files.

Rationale:

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

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `--audit-log-maxbackup` to `10` or as appropriate.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

By default, auditing is not enabled.

References:

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

2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Rotate log files on reaching 100 MB or as appropriate.

Rationale:

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

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `--audit-log-maxsize=100` to `100` or as appropriate.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

By default, auditing is not enabled.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.14 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)

Profile Applicability:

- Level 1

Description:

Do not always authorize all requests.

Rationale:

The federation apiserver, by default, allows all requests. You should restrict this behavior to only allow the authorization modes that you explicitly use in your environment. For example, if you don't use REST APIs in your environment, it is a good security best practice to switch-off that capability.

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `--authorization-mode` argument to a value other than `AlwaysAllow`

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

Only authorized requests will be served.

Default Value:

By default, `AlwaysAllow` is enabled.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml

3. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.15 Ensure that the --token-auth-file parameter is not set (Scored)

Profile Applicability:

- Level 1

Description:

Do not use token based authentication.

Rationale:

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

Audit:

Run the following command:

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

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

Remediation:

Follow the documentation and configure alternate mechanisms for authentication. Then, edit the deployment specs and remove the `--token-auth-file=<filename>` argument.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

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

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/federation-apiserver/

3. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
4. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.16 Ensure that the --service-account-lookup argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

Validate service account before validating token.

Rationale:

By default, the apiserver only verifies that the authentication token is valid. However, it 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.

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `"--service-account-lookup=true"`.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

None

Default Value:

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

References:

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

4. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
5. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.17 Ensure that the --service-account-key-file argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

By default, if no `--service-account-key-file` is specified to the federation apiserver, it uses the private key from the TLS serving certificate to verify the 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`.

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set `--service-account-key-file` argument as appropriate.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

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.

Default Value:

By default, `--service-account-key-file` argument is not set, and the private key from the TLS serving certificate is used.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver
2. https://github.com/kubernetes/kubernetes/issues/24167
3. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
4. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

3.1.18 Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

Run the following command:

```
ps -ef | grep federation-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 federation apiserver and etcd. Then, edit the deployment specs and set "`--etcd-certfile=<path/to/client-certificate-file>`" and "`--etcd-keyfile=<path/to/client-key-file>`" arguments.

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

TLS and client certificate authentication must be configured for etcd.

Default Value:

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

References:

1. https://kubernetes.io/docs/admin/federation-apiserver/
2. https://coreos.com/etcd/docs/latest/op-guide/security.html
3. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
4. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Setup TLS connection on the federation API server.

Rationale:

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

Audit:

Run the following command:

```
ps -ef | grep federation-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 federation apiserver. Then, edit the deployment specs and set `"--tls-cert-file=<path/to/tls-certificate-file>"` and `"--tls-private-key-file=<path/to/tls-key-file>"`:

```
kubectl edit deployments federation-apiserver-deployment --
namespace=federation-system
```

Impact:

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

Default Value:

By default, `--tls-cert-file` and `--tls-private-key-file` arguments are not set. If HTTPS serving is enabled, and `--tls-cert-file` and `--tls-private-key-file` are not provided, a self-signed certificate and key are generated for the public address and saved to `/var/run/kubernetes`.

References:

1. https://kubernetes.io/docs/admin/federation-apiserver
2. http://rootsquash.com/2016/05/10/securing-the-kubernetes-api/
3. https://github.com/kelseyhightower/docker-kubernetes-tls-guide
4. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-apiserver-deployment.yaml
5. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

CIS Controls:

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

Limitation and Control of Network Ports, Protocols, and Services

3.2 Federation Controller Manager

This section contains recommendations for federation controller-manager configuration.

3.2.1 Ensure that the --profiling argument is set to false (Scored)

Profile Applicability:

- Level 1

Description:

Disable profiling, if not needed.

Rationale:

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

Audit:

Run the following command:

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

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

Remediation:

Edit the deployment specs and set "`--profiling=false`":

```
kubectl edit deployments federation-controller-manager-deployment --
namespace=federation-system
```

Impact:

Profiling information would not be available.

Default Value:

By default, profiling is enabled.

References:

1. https://kubernetes.io/docs/admin/federation-controller-manager/
2. https://github.com/kubernetes/community/blob/master/contributors/devel/profiling.md
3. https://github.com/kubernetes/kubernetes/blob/master/federation/manifests/federation-controller-manager-deployment.yaml

CIS Controls:

14 Controlled Access Based on the Need to Know

Controlled Access Based on the Need to Know

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Master Node Security Configuration**		
1.1	**API Server**		
1.1.1	Ensure that the --allow-privileged argument is set to false (Scored)	☐	☐
1.1.2	Ensure that the --anonymous-auth argument is set to false (Scored)	☐	☐
1.1.3	Ensure that the --basic-auth-file argument is not set (Scored)	☐	☐
1.1.4	Ensure that the --insecure-allow-any-token argument is not set (Scored)	☐	☐
1.1.5	Ensure that the --kubelet-https argument is set to true (Scored)	☐	☐
1.1.6	Ensure that the --insecure-bind-address argument is not set (Scored)	☐	☐
1.1.7	Ensure that the --insecure-port argument is set to 0 (Scored)	☐	☐
1.1.8	Ensure that the --secure-port argument is not set to 0 (Scored)	☐	☐
1.1.9	Ensure that the --profiling argument is set to false (Scored)	☐	☐
1.1.10	Ensure that the --repair-malformed-updates argument is set to false (Scored)	☐	☐
1.1.11	Ensure that the admission control policy is not set to AlwaysAdmit (Scored)	☐	☐
1.1.12	Ensure that the admission control policy is set to AlwaysPullImages (Scored)	☐	☐
1.1.13	Ensure that the admission control policy is set to DenyEscalatingExec (Scored)	☐	☐
1.1.14	Ensure that the admission control policy is set to SecurityContextDeny (Scored)	☐	☐
1.1.15	Ensure that the admission control policy is set to NamespaceLifecycle (Scored)	☐	☐
1.1.16	Ensure that the --audit-log-path argument is set as appropriate (Scored)	☐	☐
1.1.17	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Scored)	☐	☐
1.1.18	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Scored)	☐	☐

1.1.19	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Scored)	☐	☐
1.1.20	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)	☐	☐
1.1.21	Ensure that the --token-auth-file parameter is not set (Scored)	☐	☐
1.1.22	Ensure that the --kubelet-certificate-authority argument is set as appropriate (Scored)	☐	☐
1.1.23	Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Scored)	☐	☐
1.1.24	Ensure that the --service-account-lookup argument is set to true (Scored)	☐	☐
1.1.25	Ensure that the admission control policy is set to PodSecurityPolicy (Scored)	☐	☐
1.1.26	Ensure that the --service-account-key-file argument is set as appropriate (Scored)	☐	☐
1.1.27	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Scored)	☐	☐
1.1.28	Ensure that the admission control policy is set to ServiceAccount (Scored)	☐	☐
1.1.29	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Scored)	☐	☐
1.1.30	Ensure that the --client-ca-file argument is set as appropriate (Scored)	☐	☐
1.1.31	Ensure that the --etcd-cafile argument is set as appropriate (Scored)	☐	☐
1.2	**Scheduler**		
1.2.1	Ensure that the --profiling argument is set to false (Scored)	☐	☐
1.3	**Controller Manager**		
1.3.1	Ensure that the --terminated-pod-gc-threshold argument is set as appropriate (Scored)	☐	☐
1.3.2	Ensure that the --profiling argument is set to false (Scored)	☐	☐
1.3.3	Ensure that the --insecure-experimental-approve-all-kubelet-csrs-for-group argument is not set (Scored)	☐	☐
1.3.4	Ensure that the --use-service-account-credentials argument is set to true (Scored)	☐	☐
1.3.5	Ensure that the --service-account-private-key-file argument is set as appropriate (Scored)	☐	☐
1.3.6	Ensure that the --root-ca-file argument is set as appropriate (Scored)	☐	☐
1.4	**Configuration Files**		
1.4.1	Ensure that the apiserver file permissions are set to 644 or more restrictive (Scored)	☐	☐

1.4.2	Ensure that the apiserver file ownership is set to root:root (Scored)	☐	☐
1.4.3	Ensure that the config file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.4	Ensure that the config file ownership is set to root:root (Scored)	☐	☐
1.4.5	Ensure that the scheduler file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.6	Ensure that the scheduler file ownership is set to root:root (Scored)	☐	☐
1.4.7	Ensure that the etcd.conf file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.8	Ensure that the etcd.conf file ownership is set to root:root (Scored)	☐	☐
1.4.9	Ensure that the flanneld file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.10	Ensure that the flanneld file ownership is set to root:root (Scored)	☐	☐
1.4.11	Ensure that the etcd data directory permissions are set to 700 or more restrictive (Scored)	☐	☐
1.4.12	Ensure that the etcd data directory ownership is set to etcd:etcd (Scored)	☐	☐
1.5	**etcd**		
1.5.1	Ensure that the --cert-file and --key-file arguments are set as appropriate (Scored)	☐	☐
1.5.2	Ensure that the --client-cert-auth argument is set to true (Scored)	☐	☐
1.5.3	Ensure that the --auto-tls argument is not set to true (Scored)	☐	☐
1.5.4	Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate (Scored)	☐	☐
1.5.5	Ensure that the --peer-client-cert-auth argument is set to true (Scored)	☐	☐
1.5.6	Ensure that the --peer-auto-tls argument is not set to true (Scored)	☐	☐
1.5.7	Ensure that the --wal-dir argument is set as appropriate (Scored)	☐	☐
1.5.8	Ensure that the --max-wals argument is set to 0 (Scored)	☐	☐
1.5.9	Ensure that a unique Certificate Authority is used for etcd (Not Scored)	☐	☐
1.6	**General Security Primitives**		
1.6.1	Ensure that the cluster-admin role is only used where required (Not Scored)	☐	☐
1.6.2	Create Pod Security Policies for your cluster (Not Scored)	☐	☐

1.6.3	Create administrative boundaries between resources using namespaces (Not Scored)	☐	☐
1.6.4	Create network segmentation using Network Policies (Not Scored)	☐	☐
1.6.5	Avoid using Kubernetes Secrets (Not Scored)	☐	☐
1.6.6	Ensure that the seccomp profile is set to docker/default in your pod definitions (Not Scored)	☐	☐
1.6.7	Apply Security Context to Your Pods and Containers (Not Scored)	☐	☐
1.6.8	Configure Image Provenance using ImagePolicyWebhook admission controller (Not Scored)	☐	☐
2	**Worker Node Security Configuration**		
2.1	**Kubelet**		
2.1.1	Ensure that the --allow-privileged argument is set to false (Scored)	☐	☐
2.1.2	Ensure that the --anonymous-auth argument is set to false (Scored)	☐	☐
2.1.3	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)	☐	☐
2.1.4	Ensure that the --client-ca-file argument is set as appropriate (Scored)	☐	☐
2.1.5	Ensure that the --read-only-port argument is set to 0 (Scored)	☐	☐
2.1.6	Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Scored)	☐	☐
2.1.7	Ensure that the --protect-kernel-defaults argument is set to true (Scored)	☐	☐
2.1.8	Ensure that the --make-iptables-util-chains argument is set to true (Scored)	☐	☐
2.1.9	Ensure that the --keep-terminated-pod-volumes argument is set to false (Scored)	☐	☐
2.1.10	Ensure that the --hostname-override argument is not set (Scored)	☐	☐
2.1.11	Ensure that the --event-qps argument is set to 0 (Scored)	☐	☐
2.1.12	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Scored)	☐	☐
2.1.13	Ensure that the --cadvisor-port argument is set to 0 (Scored)	☐	☐
2.2	**Configuration Files**		
2.2.1	Ensure that the config file permissions are set to 644 or more restrictive (Scored)	☐	☐
2.2.2	Ensure that the config file ownership is set to root:root (Scored)	☐	☐
2.2.3	Ensure that the kubelet file permissions are set to 644 or more restrictive (Scored)	☐	☐

2.2.4	Ensure that the kubelet file ownership is set to root:root (Scored)	☐	☐
2.2.5	Ensure that the proxy file permissions are set to 644 or more restrictive (Scored)	☐	☐
2.2.6	Ensure that the proxy file ownership is set to root:root (Scored)	☐	☐
3	**Federated Deployments**		
3.1	**Federation API Server**		
3.1.1	Ensure that the --anonymous-auth argument is set to false (Scored)	☐	☐
3.1.2	Ensure that the --basic-auth-file argument is not set (Scored)	☐	☐
3.1.3	Ensure that the --insecure-allow-any-token argument is not set (Scored)	☐	☐
3.1.4	Ensure that the --insecure-bind-address argument is not set (Scored)	☐	☐
3.1.5	Ensure that the --insecure-port argument is set to 0 (Scored)	☐	☐
3.1.6	Ensure that the --secure-port argument is not set to 0 (Scored)	☐	☐
3.1.7	Ensure that the --profiling argument is set to false (Scored)	☐	☐
3.1.8	Ensure that the admission control policy is not set to AlwaysAdmit (Scored)	☐	☐
3.1.9	Ensure that the admission control policy is set to NamespaceLifecycle (Scored)	☐	☐
3.1.10	Ensure that the --audit-log-path argument is set as appropriate (Scored)	☐	☐
3.1.11	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Scored)	☐	☐
3.1.12	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Scored)	☐	☐
3.1.13	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Scored)	☐	☐
3.1.14	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)	☐	☐
3.1.15	Ensure that the --token-auth-file parameter is not set (Scored)	☐	☐
3.1.16	Ensure that the --service-account-lookup argument is set to true (Scored)	☐	☐
3.1.17	Ensure that the --service-account-key-file argument is set as appropriate (Scored)	☐	☐
3.1.18	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Scored)	☐	☐

3.1.19	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Scored)	☐	☐
3.2	**Federation Controller Manager**		
3.2.1	Ensure that the --profiling argument is set to false (Scored)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
5-3-2017	1.0.0	Initial Release