Center for Internet Security®

CIS Kubernetes Benchmark

v1.2.0 - 10-04-2017

Table of Contents

Overview

This document provides prescriptive guidance for establishing a secure configuration posture for Kubernetes 1.8.0. To obtain the latest version of this guide, please visit http://www.cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at support@cisecurity.org.

**Special Note: **The set of configuration files mentioned anywhere throughout this benchmark document may vary according to the deployment tool and the platform. Any reference to a configuration file should be modified according to the actual configuration files used on the specific deployment.

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 1.8.

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:

Author
Pravin Goyal, Cavirin Systems, Inc
Rory McCune
Jordan Liggitt

Contributor
Puja Abbassi
Liz Rice
Eric Chiang
Jordan Rakoske GSEC, GCWN

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

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

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

1.1.2 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 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and remove the `--basic-auth-file=<filename>` parameter.

Impact:

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

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:

16.14 Encrypt/Hash All Authentication Files And Monitor Their Access
Verify that all authentication files are encrypted or hashed and that these files cannot be accessed without root or administrator privileges. Audit all access to password files in the system.

1.1.3 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 on the master node:

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

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

Remediation:

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

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

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

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:

14.2 Encrypt All Sensitive Information Over Less-trusted Networks
All communication of sensitive information over less-trusted networks should be

encrypted. Whenever information flows over a network with a lower trust level, the information should be encrypted.

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

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:

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

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

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

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:

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

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

Impact:

You need to set the API Server 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:

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

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

```
--profiling=false
```

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

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

Profile Applicability:

- Level 1

Description:

Disable fixing of malformed updates.

Rationale:

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

```
--repair-malformed-updates=false
```

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

1.1.10 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, and that its value does not include `AlwaysAdmit`.

Remediation:

Edit the API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--admission-control` parameter to a value that does not include `AlwaysAdmit`.

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

1.1.11 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--admission-control` parameter to include `AlwaysPullImages`.

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

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:

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

1.1.12 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--admission-control` parameter to a value that includes `DenyEscalatingExec`.

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

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

1.1.13 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--admission-control` parameter to include `SecurityContextDeny`.

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

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:

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

1.1.14 Ensure that the 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--admission-control` parameter to include `NamespaceLifecycle`.

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

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

Run the following command on the master node:

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

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

Remediation:

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

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

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:

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

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

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

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:

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)
Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

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

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

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:

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)
Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

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

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

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:

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)
Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

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

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

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:

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

1.1.20 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and remove the `--token-auth-file=<filename>` parameter.

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:

16.14 Encrypt/Hash All Authentication Files And Monitor Their Access
Verify that all authentication files are encrypted or hashed and that these files cannot be accessed without root or administrator privileges. Audit all access to password files in the system.

1.1.21 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--kubelet-certificate-authority` parameter to the path to the cert file for the certificate authority.

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

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:

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

1.1.22 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 itself to the kubelet's HTTPS endpoints. The requests from the apiserver are treated anonymously. You should set up certificate-based kubelet authentication to ensure that the apiserver authenticates itself to kubelets when submitting requests.

Audit:

Run the following command on the master node:

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

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

Remediation:

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

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

Impact:

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

Default Value:

By default, certificate-based kublet 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:

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

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

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

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:

16 Account Monitoring and Control
Account Monitoring and Control

1.1.24 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--admission-control` parameter to a value that includes `PodSecurityPolicy`:

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

Then restart the API Server.

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

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

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

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

1.1.26 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the etcd certificate and key file parameters.

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

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

1.1.27 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the `--admission-control` parameter to a value that includes `ServiceAccount`.

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

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:

16 Account Monitoring and Control
Account Monitoring and Control

1.1.28 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the TLS certificate and private key file parameters.

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

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:

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

1.1.29 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the client certificate authority file.

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

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:

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

1.1.30 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 API server pod specification file `/etc/kubernetes/manifests/kube-apiserver.yaml` on the master node and set the etcd certificate authority file parameter.

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

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:

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

1.1.31 Ensure that the --authorization-mode argument is set to Node (Scored)

Profile Applicability:

- Level 1

Description:

Restrict kubelet nodes to reading only objects associated with them.

Rationale:

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

Audit:

Run the following command on the master node:

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

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

Remediation:

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

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

Impact:

None

Default Value:

By default, `Node` authorization is not enabled.

References:

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

3. https://github.com/kubernetes/kubernetes/pull/46076
4. https://acotten.com/post/kube17-security

CIS Controls:

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

1.1.32 Ensure that the admission control policy is set to NodeRestriction (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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 `NodeRestriction`.

Remediation:

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

```
--admission-control=...,NodeRestriction,...
```

Impact:

None

Default Value:

By default, `NodeRestriction` is not set.

References:

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

CIS Controls:

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

1.1.33 Ensure that the --experimental-encryption-provider-config argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Encrypt etcd key-value store.

Rationale:

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

Audit:

Run the following command on the master node:

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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

1.1.34 Ensure that the encryption provider is set to aescbc (Scored)

Profile Applicability:

- Level 1

Description:

Use `aescbc` encryption provider.

Rationale:

`aescbc` is currently the strongest encryption provider, It should be preferred over other providers.

Audit:

Run the following command on the master node:

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

Get the `EncryptionConfig` file set for `--experimental-encryption-provider-config` argument. Verify that the `aescbc` encryption provider is used for all the desired `resources`.

Remediation:

Follow the Kubernetes documentation and configure a `EncryptionConfig` file. In this file, choose `aescbc` as the encryption provider.
For example,

```
kind: EncryptionConfig
apiVersion: v1
resources:
  - resources:
    - secrets
    providers:
    - aescbc:
        keys:
        - name: key1
          secret: <32-byte base64-encoded secret>
```

Impact:

None

Default Value:

By default, no encryption provider is set.

References:

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

CIS Controls:

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

1.1.35 Ensure that the admission control policy is set to EventRateLimit (Scored)

Profile Applicability:

- Level 1

Description:

Limit the rate at which the API server accepts requests.

Rationale:

Using `EventRateLimit` admission control enforces a limit on the number of events that the API Server will accept in a given time slice. In a large multi-tenant cluster, there might be a small percentage of misbehaving tenants which could have a significant impact on the performance of the cluster overall. Hence, it is recommended to limit the rate of events that the API server will accept.

Note: This is an Alpha feature in the Kubernetes 1.8 release.

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 `EventRateLimit`.

Remediation:

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

```
--admission-control=EventRateLimit
--admission-control-config-file=<path/to/configuration/file>
```

Impact:

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

Default Value:

By default, `EventRateLimit` is not set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/admin/admission-controllers
3. https://github.com/staebler/community/blob/9873b632f4d99b5d99c38c9b15fe2f8b93d0a746/contributors/design-proposals/admission_control_event_rate_limit.md

CIS Controls:

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

1.1.36 Ensure that the AdvancedAuditing argument is not set to false (Scored)

Profile Applicability:

- Level 1

Description:

Do not disable advanced auditing.

Rationale:

`AdvancedAuditing` enables a much more general API auditing pipeline, which includes support for pluggable output backends and an audit policy specifying how different requests should be audited. Additionally, this enables auditing of failed authentication, authorization and login attempts which could prove crucial for protecting your production clusters. It is thus recommended not to disable advanced auditing.

Audit:

Run the following command on the master node:

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

Verify that the `--feature-gates` argument is not set to a value that includes `AdvancedAuditing=false`. If `--feature-gates` or `AdvancedAuditing` arguments are not present, then it means that `AdvancedAuditing` is enabled and this recommendation is enforced by default.
Additionally, review the audit policy file specified in the `--audit-policy-file` argument and ensure that it is set as appropriate. At a minimum, it should have below policy set:

```
# Log all requests at the Metadata level.
rules:
- level: Metadata
```

Remediation:

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

```
--audit-policy-file=/etc/kubernetes/audit-policy.yaml
```

Impact:

You would need to rotate logs and log them centrally to avoid filling up disk space.

Default Value:

By default, `AdvancedAuditing` is set.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://github.com/kubernetes/kubernetes.github.io/blob/release-1.8/docs/tasks/debug-application-cluster/audit.md

CIS Controls:

14.6 Enforce Detailed Audit Logging For Sensitive Information
Enforce detailed audit logging for access to nonpublic data and special authentication for sensitive data.

1.1.37 Ensure that the --request-timeout argument is set as appropriate (Scored)

Profile Applicability:

- Level 1

Description:

Set global request timeout for API server requests as appropriate.

Rationale:

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

Audit:

Run the following command on the master node:

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

14.6 Enforce Detailed Audit Logging For Sensitive Information
Enforce detailed audit logging for access to nonpublic data and special authentication for sensitive data.

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

```
--profiling=false
```

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

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

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:

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

1.3.2 Ensure that the --profiling argument is set to false (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 Controller Manager pod specification file `/etc/kubernetes/manifests/kube-controller-manager.yaml` on the master node and set the below parameter.

```
--profiling=false
```

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

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

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

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

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

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

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

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

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:

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

1.3.6 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:

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

1.3.7 Ensure that the RotateKubeletServerCertificate argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

Enable kubelet server certificate rotation on controller-manager.

Rationale:

`RotateKubeletServerCertificate` causes the kubelet to both request a serving certificate after bootstrapping its client credentials and rotate the certificate as its existing credentials expire. This automated periodic rotation ensures that the there are no downtimes due to expired certificates and thus addressing availability in the CIA security triad.

Note: This recommendation only applies if you let kubelets get their certificates from the API server. In case your kubelet certificates come from an outside authority/tool (e.g. Vault) then you need to take care of rotation yourself.

Audit:

Run the following command on the master node:

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

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

Remediation:

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

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

Impact:

None

Default Value:

By default, `RotateKubeletServerCertificate` is not set.

References:

1. https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/#approval-controller
2. https://github.com/kubernetes/features/issues/267
3. https://github.com/kubernetes/kubernetes/pull/45059
4. https://kubernetes.io/docs/admin/kube-controller-manager/

CIS Controls:

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

1.4 Configuration Files

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

The API server pod specification file controls various parameters that set the behavior of the API server. You should restrict its file permissions to maintain the integrity of the file. The file should be writable 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/manifests/kube-apiserver.yaml
```

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

Remediation:

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

```
chmod 644 /etc/kubernetes/manifests/kube-apiserver.yaml
```

Impact:

None

Default Value:

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

References:

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

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

```
chmod 644 /etc/kubernetes/manifests/kube-controller-manager.yaml
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

The controller manager pod specification file controls various parameters that set the behavior of various components of the master node. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

```
chmod 644 /etc/kubernetes/manifests/kube-scheduler.yaml
```

Impact:

None

Default Value:

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

References:

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

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

The scheduler pod specification file controls various parameters that set the behavior of the `kube-scheduler` service in the master node. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

```
chmod 644 /etc/kubernetes/manifests/etcd.yaml
```

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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.

1.4.8 Ensure that the etcd pod specification file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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.

1.4.9 Ensure that the Container Network Interface file permissions are set to 644 or more restrictive (Not Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

NA

References:

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

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.

1.4.10 Ensure that the Container Network Interface file ownership is set to root:root (Not Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

NA

References:

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

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.

1.4.11 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
```

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
```

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
```

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
```

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None.

Default Value:

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

References:

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

CIS Controls:

5.1 Minimize And Sparingly Use Administrative Privileges

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

1.4.14 Ensure that the admin.conf file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None.

Default Value:

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

References:

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

CIS Controls:

5.1 Minimize And Sparingly Use Administrative Privileges

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

5.1 Minimize And Sparingly Use Administrative Privileges

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

1.4.16 Ensure that the scheduler.conf file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

5.1 Minimize And Sparingly Use Administrative Privileges

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

```
chmod 644 /etc/kubernetes/controller-manager.conf
```

Impact:

None

Default Value:

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

References:

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

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

CIS Controls:

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.

1.5 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.
Then, edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and set the below parameters.

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

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:

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

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

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

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:

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

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

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

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:

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

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.
Then, edit the etcd pod specification file `/etc/kubernetes/manifests/etcd.yaml` on the master node and set the below parameters.

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

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:

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

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

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

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:

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

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

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

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:

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

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

```
--wal-dir=</path/to/log/dir>
```

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:

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

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

```
--max-wals=0
```

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:

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

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 file used for the management of the overall Kubernetes cluster.
Run the following command on the master node:

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

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

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

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

Remediation:

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

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

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

Impact:

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

Default Value:

By default, no etcd certificate is created and used.

References:

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

CIS Controls:

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:

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

1.6.3 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:

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

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:

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

1.6.5 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:

5 Controlled Use of Administration Privileges
Controlled Use of Administration Privileges

1.6.6 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:

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

1.6.7 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

1.6.8 Configure Network policies as appropriate (Not Scored)

Profile Applicability:

- Level 2

Description:

Configure Network policies as appropriate.

Rationale:

The Network Policy API is now stable. Network policy, implemented through a network plug-in, allows users to set and enforce rules governing which pods can communicate with each other. You should leverage it as appropriate in your environment.

Audit:

Review the network policies enforced and ensure that they are suitable for your requirements.

Remediation:

Follow the Kubernetes documentation and setup network policies as appropriate.
For example, you could create a "default" isolation policy for a Namespace by creating a NetworkPolicy that selects all pods but does not allow any traffic:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector:
```

Impact:

You need to regularly maintain your network policies and design them carefully. Also, network policies v1 support depends on your CNI. Carefully choose your CNI.

Default Value:

By default, network policies are not set.

References:

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

CIS Controls:

12 Boundary Defense
Boundary Defense

1.6.9 Place compensating controls in the form of PSP and RBAC for privileged containers usage (Not Scored)

Profile Applicability:

- Level 2

Description:

Use Pod Security Policies (PSP) and RBAC authorization to mitigate the risk arising from using privileged containers.

Rationale:

A number of components used by Kubernetes clusters currently make use of privileged containers (e.g. Container Network Interface plugins). Privileged containers pose a risk to the underlying host infrastructure. You should use PSP and RBAC or other forms of authorization to mitigate the risk arising out of such privileged container usage. PSPs should be in place to restrict access to create privileged containers to specific roles only, and access to those roles should be restricted using RBAC role bindings.

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.
Additionally, review the RBAC authorization:

```
kubectl get rolebinding <role binding name>
kubectl get clusterrolebinding <cluster role binding name>
```

Remediation:

Follow Kubernetes documentation and setup PSP and RBAC authorization for your cluster.

Impact:

You need to carefully tune your PSP and RBAC authorization policy to provide minimal access to the components and various accounts.

Default Value:

By default, PSP and RBAC authorization policies are not setup.

References:

1. https://kubernetes.io/docs/admin/kube-apiserver/
2. https://kubernetes.io/docs/user-guide/security-context/
3. http://blog.kubernetes.io/2017/04/rbac-support-in-kubernetes.html
4. https://kubernetes.io/docs/concepts/policy/pod-security-policy/
5. https://kubernetes.io/docs/admin/authorization/rbac/

CIS Controls:

5 Controlled Use of Administration Privileges
Controlled Use of Administration Privileges

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

```
--allow-privileged=false
```

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

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

Impact:

You will not be able to run any privileged containers.

Default Value:

By default, privileged containers are allowed.

References:

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

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.

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

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

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

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_AUTHZ_ARGS` variable.

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

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

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

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:

Edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_AUTHZ_ARGS` variable.

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

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

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

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:

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

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

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

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

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:

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

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

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

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

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:

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

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

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

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

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:

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and remove the `--make-iptables-util-chains` argument from the `KUBELET_SYSTEM_PODS_ARGS` variable.
Based on your system, restart the `kubelet` service. For example:

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

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:

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

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

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

```
systemctl daemon-reload
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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and remove the `--hostname-override` argument from the `KUBELET_SYSTEM_PODS_ARGS` variable.
Based on your system, restart the `kubelet` service. For example:

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

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:

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

```
--event-qps=0
```

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

```
systemctl daemon-reload
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:

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameters in `KUBELET_CERTIFICATE_ARGS` variable.

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

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

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

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:

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

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 kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_CADVISOR_ARGS` variable.

```
--cadvisor-port=0
```

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

```
systemctl daemon-reload
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 `0`.

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:

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

2.1.14 Ensure that the RotateKubeletClientCertificate argument is not set to false (Scored)

Profile Applicability:

- Level 1

Description:

Enable kubelet client certificate rotation.

Rationale:

`RotateKubeletClientCertificate` causes the kubelet to rotate its client certificates by creating new CSRs as its existing credentials expire. This automated periodic rotation ensures that the there are no downtimes due to expired certificates and thus addressing availability in the CIA security triad.

Note: This recommendation only applies if you let kubelets get their certificates from the API server. In case your kubelet certificates come from an outside authority/tool (e.g. Vault) then you need to take care of rotation yourself.

Audit:

Run the following command on each node:

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

Verify that `RotateKubeletClientCertificate` argument if exists is not set to `false`.

Remediation:

Edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and remove the `--feature-gates=RotateKubeletClientCertificate=false` argument from the `KUBELET_CERTIFICATE_ARGS` variable.
Based on your system, restart the `kubelet` service. For example:

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

Impact:

None

Default Value:

By default, kubelet client certificate rotation is enabled.

References:

1. https://github.com/kubernetes/kubernetes/pull/41912
2. https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/#kubelet-configuration

CIS Controls:

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

2.1.15 Ensure that the RotateKubeletServerCertificate argument is set to true (Scored)

Profile Applicability:

- Level 1

Description:

Enable kubelet server certificate rotation.

Rationale:

`RotateKubeletServerCertificate` causes the kubelet to both request a serving certificate after bootstrapping its client credentials and rotate the certificate as its existing credentials expire. This automated periodic rotation ensures that the there are no downtimes due to expired certificates and thus addressing availability in the CIA security triad.

Note: This recommendation only applies if you let kubelets get their certificates from the API server. In case your kubelet certificates come from an outside authority/tool (e.g. Vault) then you need to take care of rotation yourself.

Audit:

Run the following command on each node:

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

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

Remediation:

Edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_CERTIFICATE_ARGS` variable.

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

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

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

Impact:

None

Default Value:

By default, kubelet server certificate rotation is disabled.

References:

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

CIS Controls:

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

2.2 Configuration Files

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

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

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.conf
```

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

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.

2.2.2 Ensure that the kubelet.conf file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `kubelet.conf` file is the kubeconfig file for the node, and controls various parameters that set the behavior and identity of the worker node. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

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.conf
```

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

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

CIS Controls:

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

The `kubelet` service file controls various parameters that set the behavior of the `kubelet` service in the worker node. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

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/systemd/system/kubelet.service.d/10-kubeadm.conf
```

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

Remediation:

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

```
chown root:root /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Impact:

None

Default Value:

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

References:

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

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

CIS Controls:

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.

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

Profile Applicability:

- Level 1

Description:

If `kube-proxy` is running, ensure that the proxy kubeconfig file has permissions of `644` or more restrictive.

Rationale:

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

Audit:

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

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

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

```
stat -c %a <proxy kubeconfig file>
```

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 <proxy kubeconfig file>
```

Impact:

None

Default Value:

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

References:

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

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

The kubeconfig file for `kube-proxy` controls various parameters for the `kube-proxy` service in the worker node. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Audit:

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

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

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

```
stat -c %U:%G <proxy kubeconfig file>
```

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

Remediation:

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

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

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:

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.

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Audit:

Run the following command:

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

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

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

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

Remediation:

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

```
chmod 644 <filename>
```

Impact:

None

Default Value:

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

References:

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

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.

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

2.2.8 Ensure that the client certificate authorities file ownership is set to root:root (Scored)

Profile Applicability:

- Level 1

Description:

Ensure that the certificate authorities file ownership is set to root:root.

Rationale:

The certificate authorities file controls the authorities used to validate API requests. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Audit:

Run the following command:

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

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

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

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

Remediation:

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

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

Impact:

None

Default Value:

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

References:

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

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.

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:

16.14 Encrypt/Hash All Authentication Files And Monitor Their Access
Verify that all authentication files are encrypted or hashed and that these files cannot be accessed without root or administrator privileges. Audit all access to password files in the system.

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:

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

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:

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

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:

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

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:

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

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:

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)
Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

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:

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)
Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

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:

6.3 Ensure Audit Logging Systems Are Not Subject To Loss (i.e. rotation/archive)
Ensure that all systems that store logs have adequate storage space for the logs generated on a regular basis, so that log files will not fill up between log rotation intervals. The logs must be archived and digitally signed on a periodic basis.

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:

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

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:

16.14 Encrypt/Hash All Authentication Files And Monitor Their Access
Verify that all authentication files are encrypted or hashed and that these files cannot be accessed without root or administrator privileges. Audit all access to password files in the system.

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:

16 Account Monitoring and Control
Account Monitoring and Control

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:

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

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 --anonymous-auth argument is set to false (Scored)	☐	☐
1.1.2	Ensure that the --basic-auth-file argument is not set (Scored)	☐	☐
1.1.3	Ensure that the --insecure-allow-any-token argument is not set (Scored)	☐	☐
1.1.4	Ensure that the --kubelet-https argument is set to true (Scored)	☐	☐
1.1.5	Ensure that the --insecure-bind-address argument is not set (Scored)	☐	☐
1.1.6	Ensure that the --insecure-port argument is set to 0 (Scored)	☐	☐
1.1.7	Ensure that the --secure-port argument is not set to 0 (Scored)	☐	☐
1.1.8	Ensure that the --profiling argument is set to false (Scored)	☐	☐
1.1.9	Ensure that the --repair-malformed-updates argument is set to false (Scored)	☐	☐
1.1.10	Ensure that the admission control policy is not set to AlwaysAdmit (Scored)	☐	☐
1.1.11	Ensure that the admission control policy is set to AlwaysPullImages (Scored)	☐	☐
1.1.12	Ensure that the admission control policy is set to DenyEscalatingExec (Scored)	☐	☐
1.1.13	Ensure that the admission control policy is set to SecurityContextDeny (Scored)	☐	☐
1.1.14	Ensure that the admission control policy is set to NamespaceLifecycle (Scored)	☐	☐
1.1.15	Ensure that the --audit-log-path argument is set as appropriate (Scored)	☐	☐
1.1.16	Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Scored)	☐	☐
1.1.17	Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Scored)	☐	☐
1.1.18	Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Scored)	☐	☐

1.1.19	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)	☐	☐
1.1.20	Ensure that the --token-auth-file parameter is not set (Scored)	☐	☐
1.1.21	Ensure that the --kubelet-certificate-authority argument is set as appropriate (Scored)	☐	☐
1.1.22	Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Scored)	☐	☐
1.1.23	Ensure that the --service-account-lookup argument is set to true (Scored)	☐	☐
1.1.24	Ensure that the admission control policy is set to PodSecurityPolicy (Scored)	☐	☐
1.1.25	Ensure that the --service-account-key-file argument is set as appropriate (Scored)	☐	☐
1.1.26	Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Scored)	☐	☐
1.1.27	Ensure that the admission control policy is set to ServiceAccount (Scored)	☐	☐
1.1.28	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Scored)	☐	☐
1.1.29	Ensure that the --client-ca-file argument is set as appropriate (Scored)	☐	☐
1.1.30	Ensure that the --etcd-cafile argument is set as appropriate (Scored)	☐	☐
1.1.31	Ensure that the --authorization-mode argument is set to Node (Scored)	☐	☐
1.1.32	Ensure that the admission control policy is set to NodeRestriction (Scored)	☐	☐
1.1.33	Ensure that the --experimental-encryption-provider-config argument is set as appropriate (Scored)	☐	☐
1.1.34	Ensure that the encryption provider is set to aescbc (Scored)	☐	☐
1.1.35	Ensure that the admission control policy is set to EventRateLimit (Scored)	☐	☐
1.1.36	Ensure that the AdvancedAuditing argument is not set to false (Scored)	☐	☐
1.1.37	Ensure that the --request-timeout 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 --use-service-account-credentials argument is set to true (Scored)	☐	☐
1.3.4	Ensure that the --service-account-private-key-file argument is set as appropriate (Scored)	☐	☐
1.3.5	Ensure that the --root-ca-file argument is set as appropriate (Scored)	☐	☐
1.3.6	Apply Security Context to Your Pods and Containers (Not Scored)	☐	☐
1.3.7	Ensure that the RotateKubeletServerCertificate argument is set to true (Scored)	☐	☐
1.4	**Configuration Files**		
1.4.1	Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.2	Ensure that the API server pod specification file ownership is set to root:root (Scored)	☐	☐
1.4.3	Ensure that the controller manager pod specification file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.4	Ensure that the controller manager pod specification file ownership is set to root:root (Scored)	☐	☐
1.4.5	Ensure that the scheduler pod specification file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.6	Ensure that the scheduler pod specification file ownership is set to root:root (Scored)	☐	☐
1.4.7	Ensure that the etcd pod specification file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.8	Ensure that the etcd pod specification file ownership is set to root:root (Scored)	☐	☐
1.4.9	Ensure that the Container Network Interface file permissions are set to 644 or more restrictive (Not Scored)	☐	☐
1.4.10	Ensure that the Container Network Interface file ownership is set to root:root (Not 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)	☐	☐
1.4.13	Ensure that the admin.conf file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.14	Ensure that the admin.conf file ownership is set to root:root (Scored)	☐	☐
1.4.15	Ensure that the scheduler.conf file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.16	Ensure that the scheduler.conf file ownership is set to root:root (Scored)	☐	☐

1.4.17	Ensure that the controller-manager.conf file permissions are set to 644 or more restrictive (Scored)	☐	☐
1.4.18	Ensure that the controller-manager.conf file ownership is set to root:root (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	Ensure that the seccomp profile is set to docker/default in your pod definitions (Not Scored)	☐	☐
1.6.6	Apply Security Context to Your Pods and Containers (Not Scored)	☐	☐
1.6.7	Configure Image Provenance using ImagePolicyWebhook admission controller (Not Scored)	☐	☐
1.6.8	Configure Network policies as appropriate (Not Scored)	☐	☐
1.6.9	Place compensating controls in the form of PSP and RBAC for privileged containers usage (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)	☐	☐
2.1.14	Ensure that the RotateKubeletClientCertificate argument is not set to false (Scored)	☐	☐
2.1.15	Ensure that the RotateKubeletServerCertificate argument is set to true (Scored)	☐	☐
2.2	**Configuration Files**		
2.2.1	Ensure that the kubelet.conf file permissions are set to 644 or more restrictive (Scored)	☐	☐
2.2.2	Ensure that the kubelet.conf file ownership is set to root:root (Scored)	☐	☐
2.2.3	Ensure that the kubelet service file permissions are set to 644 or more restrictive (Scored)	☐	☐
2.2.4	Ensure that the kubelet service file ownership is set to root:root (Scored)	☐	☐
2.2.5	Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive (Scored)	☐	☐
2.2.6	Ensure that the proxy kubeconfig file ownership is set to root:root (Scored)	☐	☐
2.2.7	Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Scored)	☐	☐
2.2.8	Ensure that the client certificate authorities 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
5/15/2017	1.0.0	Initial Release
7/13/2017	1.1.0	NEW - 1.1.32 - Ensure that the --authorization-mode argument is set to Node. Ticket # 5251
7/13/2017	1.1.0	NEW - 1.1.33 Ensure that the admission control policy is set to NodeRestriction. Ticket # 5270
7/13/2017	1.1.0	NEW - 1.1.34 Ensure that the --experimental-encryption-provider-config argument is set as appropriate. Ticket # 5274
7/13/2017	1.1.0	NEW - 1.1.35 Ensure that the encryption provider is set to aescbc. Ticket # 5276
7/13/2017	1.1.0	NEW - 1.3.7 Ensure that the RotateKubeletServerCertificate argument is set to true. Ticket #5272
7/13/2017	1.1.0	NEW - 1.6.8 Configure Network policies as appropriate. Ticket # 5061
7/13/2017	1.1.0	NEW - 2.1.14 Ensure that the RotateKubeletClientCertificate argument is set to true. Ticket 5271
7/13/2017	1.1.0	NEW - 2.1.15 Ensure that the RotateKubeletServerCertificate argument is set to true. Ticket 5272

7/13/2017	1.1.0	NEW - 2.2.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive. Ticket # 5243
7/13/2017	1.1.0	NEW - 2.2.8 Ensure that the client certificate authorities file ownership is set to root:root. Ticket # 5244
7/13/2017	1.1.0	REMOVED - 1.3.3 Ensure that the --insecure-experimental-approve-all-kubelet-csrs-for-group argument is not set. Ticket # 5197
7/13/2017	1.1.0	REMOVED - 1.6.5 Avoid using Kubernetes Secrets. Ticket # 5273
10/4/2017	1.2.0	NEW - 1.1.35 Ensure that the admission control policy is set to EventRateLimit. Ticket #5487
10/4/2017	1.2.0	NEW - 1.1.36 Ensure that the AdvancedAuditing argument is set to true. Ticket # 5488
10/4/2017	1.2.0	NEW - 1.1.37 Ensure that the --request-timeout argument is set as appropriate. Ticket 5518
10/4/2017	1.2.0	NEW - 1.4.13 Ensure that the admin.conf file permissions are set to 644 or more restrictive
10/4/2017	1.2.0	NEW - 1.4.14 Ensure that the admin.conf file ownership is set to root:root
10/4/2017	1.2.0	NEW - 1.4.15 Ensure that the scheduler.conf file permissions are set to 644 or more restrictive

10/4/2017	1.2.0	NEW - 1.4.16 Ensure that the scheduler.conf file ownership is set to root:root
10/4/2017	1.2.0	NEW - 1.4.17 Ensure that the controller-manager.conf file permissions are set to 644 or more restrictive
10/4/2017	1.2.0	NEW - 1.4.18 Ensure that the controller-manager.conf file ownership is set to root:root
10/4/2017	1.2.0	MODIFY - 1.1.1 Ensure that the --allow-privileged argument is set to false moved to 1.6.9 Refactored the Remediation procedure for all recommendations to follow kubeadm scheme of things