CIS Amazon Elastic Kubernetes Service (EKS) Benchmark

v1.8.0 - 10-22-2025

Terms of Use

Table of Contents

Overview

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

- Monitoring the base operating system and applications for vulnerabilities and quickly updating with the latest security patches.
- End-point protection (Antivirus software, Endpoint Detection and Response (EDR), etc.).
- Logging and monitoring user and system activity.

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

Important Usage Information

All Benchmarks are available free for non-commercial use from the CIS Website. They can be used to manually assess and remediate systems and applications. In lieu of manual assessment and remediation, there are several tools available to assist with assessment:

- CIS Configuration Assessment Tool (CIS-CAT® Pro Assessor)
- CIS Benchmarks™ Certified 3rd Party Tooling

These tools make the hardening process much more scalable for large numbers of systems and applications.

NOTE: Some tooling focuses only on the Benchmark Recommendations that can be fully automated (skipping ones marked **Manual**). It is important that ***ALL*** Recommendations (**Automated** and **Manual**) be addressed since all are important for properly securing systems and are typically in scope for audits.

Key Stakeholders

Cybersecurity is a collaborative effort, and cross functional cooperation is imperative within an organization to discuss, test, and deploy Benchmarks in an effective and efficient way. The Benchmarks are developed to be best practice configuration guidelines applicable to a wide range of use cases. In some organizations, exceptions to specific Recommendations will be needed, and this team should work to prioritize the problematic Recommendations based on several factors like risk, time, cost, and labor. These exceptions should be properly categorized and documented for auditing purposes.

Apply the Correct Version of a Benchmark

Benchmarks are developed and tested for a specific set of products and versions and applying an incorrect Benchmark to a system can cause the resulting pass/fail score to be incorrect. This is due to the assessment of settings that do not apply to the target systems. To assure the correct Benchmark is being assessed:

- **Deploy the Benchmark applicable to the way settings are managed in the environment:** An example of this is the Microsoft Windows family of Benchmarks, which have separate Benchmarks for Group Policy, Intune, and Stand-alone systems based upon how system management is deployed. Applying the wrong Benchmark in this case will give invalid results.

- **Use the most recent version of a Benchmark**: This is true for all Benchmarks, but especially true for cloud technologies. Cloud technologies change frequently and using an older version of a Benchmark may have invalid methods for auditing and remediation.

Exceptions

The guidance items in the Benchmarks are called recommendations and not requirements, and exceptions to some of them are expected and acceptable. The Benchmarks strive to be a secure baseline, or starting point, for a specific technology, with known issues identified during Benchmark development are documented in the Impact section of each Recommendation. In addition, organizational, system specific requirements, or local site policy may require changes as well, or an exception to a Recommendation or group of Recommendations (e.g. A Benchmark could Recommend that a Web server not be installed on the system, but if a system's primary purpose is to function as a Webserver, there should be a documented exception to this Recommendation for that specific server).

In the end, exceptions to some Benchmark Recommendations are common and acceptable, and should be handled as follows:

- The reasons for the exception should be reviewed cross-functionally and be well documented for audit purposes.
- A plan should be developed for mitigating, or eliminating, the exception in the future, if applicable.
- If the organization decides to accept the risk of this exception (not work toward mitigation or elimination), this should be documented for audit purposes.

It is the responsibility of the organization to determine their overall security policy, and which settings are applicable to their unique needs based on the overall risk profile for the organization.

Remediation

CIS has developed Build Kits for many technologies to assist in the automation of hardening systems. Build Kits are designed to correspond to Benchmark's "Remediation" section, which provides the manual remediation steps necessary to make that Recommendation compliant to the Benchmark.

When remediating systems (changing configuration settings on deployed systems as per the Benchmark's Recommendations), please approach this with caution and test thoroughly.

The following is a reasonable remediation approach to follow:

- CIS Build Kits, or internally developed remediation methods should never be applied to production systems without proper testing.
- Proper testing consists of the following:
  - Understand the configuration (including installed applications) of the targeted systems. Various parts of the organization may need different configurations (e.g., software developers vs standard office workers).
  - Read the Impact section of the given Recommendation to help determine if there might be an issue with the targeted systems.
  - Test the configuration changes with representative lab system(s). If issues arise during testing, they can be resolved prior to deploying to any production systems.
  - When testing is complete, initially deploy to a small sub-set of production systems and monitor closely for issues. If there are issues, they can be resolved prior to deploying more broadly.
  - When the initial deployment above is completes successfully, iteratively deploy to additional systems and monitor closely for issues. Repeat this process until the full deployment is complete.

Summary

Using the Benchmarks Certified tools, working as a team with key stakeholders, being selective with exceptions, and being careful with remediation deployment, it is possible to harden large numbers of deployed systems in a cost effective, efficient, and safe manner.

NOTE: As previously stated, the PDF versions of the CIS Benchmarks™ are available for free, non-commercial use on the CIS Website. All other formats of the CIS Benchmarks™ (MS Word, Excel, and Build Kits) are available for CIS SecureSuite® members.

CIS-CAT® Pro is also available to CIS SecureSuite® members.

Target Technology Details

This document provides prescriptive guidance for running Amazon Elastic Kubernetes Service (EKS) following recommended security controls. This benchmark only includes controls which can be modified by an end user of Amazon EKS.

To obtain the latest version of this guide, please visit www.cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at support@cisecurity.org.

Intended Audience

This document is intended for cluster administrators, security specialists, auditors, and any personnel who plan to develop, deploy, assess, or secure solutions that incorporate Amazon EKS using managed and self-managed nodes.

Customers using Amazon EKS on AWS Fargate are not responsible for node management. Hence, this document is not scoped for Amazon EKS on AWS Fargate customers.

Consensus Guidance

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

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

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
`Monospace font`	Used for inline code, commands, UI/Menu selections or examples. Text should be interpreted exactly as presented.
`<Monospace font in brackets>`	Text set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to reference other relevant settings, CIS Benchmarks and/or Benchmark Communities. Also, used to denote the title of a book, article, or other publication.
Bold font	Additional information or caveats things like **Notes**, **Warnings**, or **Cautions** (usually just the word itself and the rest of the text normal).

Recommendation Definitions

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

Title

Concise description for the recommendation's intended configuration.

Assessment Status

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

Automated

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

Manual

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

Profile

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

Description

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

Rationale Statement

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

Impact Statement

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

Audit Procedure

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

Remediation Procedure

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

Default Value

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

References

Additional documentation relative to the recommendation.

CIS Critical Security Controls® (CIS Controls®)

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

Additional Information

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

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

  Level 1 Configuration Profile

- **Level 2**

  Extends Level 1

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:

Thanks to Authors: Paavan Mistry and Randall Mowen
with special thanks to contributors: Rory MCcune and Tony Wilwerding

Authors:

Paavan Mistry
Randall Mowen

Editor:
Randall Mowen

Contributors
Mark Larinde
Rory MCcune
Tony Wilwerding
James Stocks
Daniel Burns
Joe Bowbeer

Recommendations

1 Control Plane Components

Security is a shared responsibility between AWS and the Amazon EKS customer. The shared responsibility model describes this as security of the cloud and security in the cloud:

Security of the cloud – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. For Amazon EKS, AWS is responsible for the Kubernetes control plane, which includes the control plane nodes and etcd database. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon EKS, see AWS Services in Scope by Compliance Program.

Security in the cloud – Your responsibility includes the following areas.

- The security configuration of the data plane, including the configuration of the security groups that allow traffic to pass from the Amazon EKS control plane into the customer VPC
- The configuration of the worker nodes and the containers themselves
- The worker node guest operating system (including updates and security patches)
    - Amazon EKS follows the shared responsibility model for CVEs and security patches on managed node groups. Because managed nodes run the Amazon EKS-optimized AMIs, Amazon EKS is responsible for building patched versions of these AMIs when bugs or issues are reported and we are able to publish a fix. However, customers are responsible for deploying these patched AMI versions to your managed node groups.
- Other associated application software:
    - Setting up and managing network controls, such as firewall rules
    - Managing platform-level identity and access management, either with or in addition to IAM
- The sensitivity of your data, your company's requirements, and applicable laws and regulations

AWS is responsible for securing the control plane, though you might be able to configure certain options based on your requirements. Section 2 of this Benchmark addresses these configurations.

2 Control Plane Configuration

This section contains recommendations for Amazon EKS control plane logging configuration. Customers are able to configure logging for control plane in Amazon EKS.

2.1 Logging

2.1.1 Enable audit Logs (Automated)

Profile Applicability:

- Level 1

Description:

Control plane logs provide visibility into operation of the EKS Control plane component systems. The API server audit logs record all accepted and rejected requests in the cluster. When enabled via EKS configuration the control plane logs for a cluster are exported to a CloudWatch Log Group for persistence.

Rationale:

Audit logs enable visibility into all API server requests from authentic and anonymous sources. Stored log data can be analyzed manually or with tools to identify and understand anomalous or negative activity and lead to intelligent remediations.

Impact:

Enabling control plane logs, including API server audit logs for Amazon EKS clusters, significantly strengthens our security posture by providing detailed visibility into all API requests, thereby reducing our attack surface. By exporting these logs to a CloudWatch Log Group, we ensure persistent storage and facilitate both manual and automated analysis to quickly identify and remediate anomalous activities. While this configuration might slightly impact usability or performance due to the overhead of logging, the enhanced security and compliance benefits far outweigh these drawbacks, making it a critical component of our security strategy.

Audit:

From Console:

1. For each EKS Cluster in each region;
2. Go to 'Amazon EKS' > 'Clusters' > 'CLUSTER_NAME' > 'Configuration' > 'Logging'.
3. This will show the control plane logging configuration:

```
API server: Enabled / Disabled
Audit: Enabled / Disabled
Authenticator: Enabled / Disabled
Controller manager: Enabled / Disabled
Scheduler: Enabled / Disabled
```

4. Ensure that all options are set to 'Enabled'.

From CLI:

```
# For each EKS Cluster in each region;
export CLUSTER_NAME=<your cluster name>
export REGION_CODE=<your region_code>
aws eks describe-cluster --name ${CLUSTER_NAME} --region ${REGION_CODE} --
query 'cluster.logging.clusterLogging'
```

Remediation:

From Console:

1. For each EKS Cluster in each region;
2. Go to 'Amazon EKS' > 'Clusters' > '' > 'Configuration' > 'Logging'.
3. Click 'Manage logging'.
4. Ensure that all options are toggled to 'Enabled'.

```
API server: Enabled
Audit: Enabled
Authenticator: Enabled
Controller manager: Enabled
Scheduler: Enabled
```

5. Click 'Save Changes'.

From CLI:

```
# For each EKS Cluster in each region;
aws eks update-cluster-config \
    --region '${REGION_CODE}' \
    --name '${CLUSTER_NAME}' \
    --logging
'{"clusterLogging":[{"types":["api","audit","authenticator","controllerManage
r","scheduler"],"enabled":true}]}'
```

Default Value:

Control Plane Logging is disabled by default.

```
API server: Disabled
Audit: Disabled
Authenticator: Disabled
Controller manager: Disabled
Scheduler: Disabled
```

References:

1. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
2. https://aws.github.io/aws-eks-best-practices/detective/
3. https://docs.aws.amazon.com/eks/latest/userguide/control-plane-logs.html
4. https://docs.aws.amazon.com/eks/latest/userguide/logging-using-cloudtrail.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.1 Establish and Maintain an Audit Log Management Process Establish and maintain an audit log management process that defines the enterprise's logging requirements. At a minimum, address the collection, review, and retention of audit logs for enterprise assets. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.	●	●	●
v7	6.2 Activate audit logging Ensure that local logging has been enabled on all systems and networking devices.	●	●	●
v7	6.3 Enable Detailed Logging Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.		●	●

2.1.2 Ensure audit logs are collected and managed (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that audit logs are collected and managed in accordance with the enterprise's audit log management process across all Kubernetes components.

Rationale:

Audit logs provide visibility into the activities occurring within a Kubernetes cluster, enabling the detection and investigation of security incidents and policy violations. Proper collection and management of audit logs are essential for maintaining an audit trail and ensuring compliance with security policies.

Impact:

Implementing comprehensive audit logging may require additional storage and processing resources. Care must be taken to ensure that logs are properly secured and managed to avoid any potential security risks associated with log data.

Audit:

1. Verify audit logging is enabled for Kubernetes components:

```
kubectl get --raw /api/v1/nodes/${NODE_NAME}/proxy/configz | jq
'.kubeletConfig.auditPolicy'
```

2. Ensure the audit logs are being collected and sent to a centralized logging system:

```
kubectl get --raw /api/v1/nodes/${NODE_NAME}/proxy/stats/summary | jq
'.auditLogs'
```

3. Verify that the audit logs are being monitored and managed according to the enterprise's audit log management process.

Remediation:

1. Create or update the audit-policy.yaml to specify the audit logging configuration:

```
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
  - level: Metadata
    resources:
      - group: ""
        resources: ["pods"]
```

2. Apply the audit policy configuration to the cluster:

```
kubectl apply -f <path-to-audit-policy>.yaml
```

3. Ensure audit logs are forwarded to a centralized logging system like CloudWatch, Elasticsearch, or another log management solution:

```
kubectl create configmap cluster-audit-policy --from-file=audit-policy.yaml -
n kube-system
kubectl apply -f - <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: audit-logging
  namespace: kube-system
spec:
  containers:
  - name: audit-log-forwarder
    image: my-log-forwarder-image
    volumeMounts:
    - mountPath: /etc/kubernetes/audit
      name: audit-config
  volumes:
  - name: audit-config
    configMap:
      name: cluster-audit-policy
EOF
```

Default Value:

By default, Kubernetes does not enable detailed audit logging. Configuration is required to enable and manage audit logs.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.1 Establish and Maintain an Audit Log Management Process Establish and maintain an audit log management process that defines the enterprise's logging requirements. At a minimum, address the collection, review, and retention of audit logs for enterprise assets. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.	●	●	●
v7	6.2 Activate audit logging Ensure that local logging has been enabled on all systems and networking devices.	●	●	●
v7	6.3 Enable Detailed Logging Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.		●	●

3 Worker Nodes

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

3.1 Worker Node Configuration Files

This section covers recommendations for configuration files on Amazon EKS worker nodes.

3.1.1 Ensure that the kubeconfig file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

If kubelet is running, and if it is configured by a kubeconfig file, ensure that the proxy kubeconfig file has permissions of 644 or more restrictive.

Rationale:

The `kubelet` kubeconfig file controls various parameters 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.

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

Impact:

Ensuring that the kubeconfig file permissions are set to 644 or more restrictive significantly strengthens the security posture of the Kubernetes environment by preventing unauthorized modifications. This restricts write access to the kubeconfig file, ensuring only administrators can alter crucial kubelet configurations, thereby reducing the risk of malicious alterations that could compromise the cluster's integrity.

However, this configuration may slightly impact usability, as it limits the ability for non-administrative users to make quick adjustments to the kubelet settings. Administrators will need to balance security needs with operational flexibility, potentially requiring adjustments to workflows for managing kubelet configurations.

Audit:

Method 1

SSH to the worker nodes

To check to see if the Kubelet Service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate kubeconfig file:

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

The output of the above command should return something similar to `--kubeconfig /var/lib/kubelet/kubeconfig` which is the location of the kubeconfig file.

Run this command to obtain the kubeconfig file permissions:

```
stat -c %a /var/lib/kubelet/kubeconfig
```

The output of the above command gives you the kubeconfig file's permissions.

Verify that if a file is specified and it exists, the permissions are 644 or more restrictive.

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file permissions on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the permission level of the file:

```
ls -l /host/var/lib/kubelet/kubeconfig
```

Verify that if a file is specified and it exists, 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 <kubeconfig file>
```

Default Value:

See the AWS EKS documentation for the default value.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

3.1.2 Ensure that the kubelet kubeconfig file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Impact:

None

Audit:

Method 1

SSH to the worker nodes

To check to see if the Kubelet Service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate kubeconfig file:

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

The output of the above command should return something similar to `--kubeconfig /var/lib/kubelet/kubeconfig` which is the location of the kubeconfig file.

Run this command to obtain the kubeconfig file ownership:

```
stat -c %U:%G /var/lib/kubelet/kubeconfig
```

The output of the above command gives you the kubeconfig file's ownership. Verify that the ownership is set to `root:root`.

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file ownership on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the ownership of the file:

```
ls -l /host/var/lib/kubelet/kubeconfig
```

The output of the above command gives you the kubeconfig file's ownership. Verify that the ownership is set to `root:root`.

Remediation:

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

For example,

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

Default Value:

See the AWS EKS documentation for the default value.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

3.1.3 Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that if the kubelet refers to a configuration file with the `--config` argument, that file has permissions of 644 or more restrictive.

Rationale:

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

Impact:

None.

Audit:

Method 1

First, SSH to the relevant worker node:

To check to see if the Kubelet Service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate Kubelet config file:

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

The output of the above command should return something similar to `--config /etc/kubernetes/kubelet/config.json` which is the location of the Kubelet config file.

Run the following command:

```
stat -c %a /etc/kubernetes/kubelet/config.json
```

The output of the above command is the Kubelet config file's permissions. Verify that the permissions are `644` or more restrictive.

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file permissions on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the permission level of the file:

```
ls -l /host/etc/kubernetes/kubelet/config.json
```

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

Remediation:

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

```
chmod 644 /etc/kubernetes/kubelet/config.json
```

Default Value:

See the AWS EKS documentation for the default value.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

3.1.4 Ensure that the kubelet configuration file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that if the kubelet refers to a configuration file with the `--config` argument, that file is owned by root:root.

Rationale:

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

Impact:

None

Audit:

Method 1

First, SSH to the relevant worker node:

To check to see if the Kubelet Service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate Kubelet config file:

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

The output of the above command should return something similar to `--config /etc/kubernetes/kubelet/config.json` which is the location of the Kubelet config file.

Run the following command:

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

The output of the above command is the Kubelet config file's ownership. Verify that the ownership is set to `root:root`

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file ownership on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the ownership of the file:

```
ls -l /etc/kubernetes/kubelet/config.json
```

The output of the above command gives you the azure.json file's ownership. Verify that the ownership is set to `root:root`.

Remediation:

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

```
chown root:root /etc/kubernetes/kubelet/config.json
```

Default Value:

See the AWS EKS documentation for the default value.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.	●	●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

3.2 Kubelet

Kubelets can accept configuration via a configuration file and in some cases via command line arguments. It is important to note that parameters provided as command line arguments will override their counterpart parameters in the configuration file (see `--config` details in the Kubelet CLI Reference for more info, where you can also find out which configuration parameters can be supplied as a command line argument).

With this in mind, it is important to check for the existence of command line arguments as well as configuration file entries when auditing Kubelet configuration.

Firstly, SSH to each node and execute the following command to find the Kubelet process:

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

The output of the above command provides details of the active Kubelet process, from which we can see the command line arguments provided to the process. Also note the location of the configuration file, provided with the `--config` argument, as this will be needed to verify configuration. The file can be viewed with a command such as `more` or `less`, like so:

```
sudo less /path/to/kubelet-config.json
```

This config file could be in JSON or YAML format depending on your distribution.

3.2.1 Ensure that the Anonymous Auth is Not Enabled (Automated)

Profile Applicability:

- Level 1

Description:

Disable anonymous requests to the Kubelet server.

Rationale:

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

Impact:

This configuration might have a slight impact on usability for users who rely on anonymous access for certain functions or quick troubleshooting. Additionally, there might be a minimal performance overhead due to the added authentication steps for each request.

Audit:

Audit Method 1:

Kubelets can accept configuration via a configuration file and in some cases via command line arguments. It is important to note that parameters provided as command line arguments will override their counterpart parameters in the configuration file (see `--config` details in the [Kubelet CLI Reference] for more info, where you can also find out which configuration parameters can be supplied as a command line argument).

With this in mind, it is important to check for the existence of command line arguments as well as configuration file entries when auditing Kubelet configuration.

Firstly, SSH to each node and execute the following command to find the Kubelet process:

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

The output of the above command provides details of the active Kubelet process, from which we can see the command line arguments provided to the process. Also note the location of the configuration file, provided with the `--config` argument, as this will be needed to verify configuration. The file can be viewed with a command such as `more` or `less`, like so:

```
sudo less /path/to/kubelet-config.json
```

Verify that Anonymous Authentication is not enabled. This may be configured as a command line argument to the kubelet service with `--anonymous-auth=false` or in the kubelet configuration file via `"authentication": { "anonymous": { "enabled": false }`.

Audit Method 2:

It is also possible to review the running configuration of a Kubelet via the /configz endpoint of the Kubernetes API. This can be achieved using `kubectl` to proxy your requests to the API.

Discover all nodes in your cluster by running the following command:

```
kubectl get nodes
```

Next, initiate a proxy with `kubectl` on a local port of your choice. In this example we will use 8080:

```
kubectl proxy --port=8080
```

With this running, in a separate terminal run the following command for each node:

```
export NODE_NAME=my-node-name
curl http://localhost:8080/api/v1/nodes/${NODE_NAME}/proxy/configz
```

The curl command will return the API response which will be a JSON formatted string representing the Kubelet configuration.

Verify that Anonymous Authentication is not enabled checking that `"authentication": { "anonymous": { "enabled": false }` is in the API response.

Remediation:

Remediation Method 1:

If configuring via the Kubelet config file, you first need to locate the file.

To do this, SSH to each node and execute the following command to find the kubelet process:

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

The output of the above command provides details of the active kubelet process, from which we can see the location of the configuration file provided to the kubelet service with the `--config` argument. The file can be viewed with a command such as `more` or `less`, like so:

```
sudo less /path/to/kubelet-config.json
```

Disable Anonymous Authentication by setting the following parameter:

```
"authentication": { "anonymous": { "enabled": false } }
```

Remediation Method 2:

If using executable arguments, edit the kubelet service file on each worker node and ensure the below parameters are part of the `KUBELET_ARGS` variable string.

For systems using `systemd`, such as the Amazon EKS Optimised Amazon Linux or Bottlerocket AMIs, then this file can be found at `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf`. Otherwise, you may need to look up documentation for your chosen operating system to determine which service manager is configured:

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

For Both Remediation Steps:

Based on your system, restart the `kubelet` service and check the service status.

The following example is for operating systems using `systemd`, such as the Amazon EKS Optimised Amazon Linux or Bottlerocket AMIs, and invokes the `systemctl` command. If `systemctl` is not available then you will need to look up documentation for your chosen operating system to determine which service manager is configured:

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

Default Value:

See the EKS documentation for the default value.

References:

1. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
2. https://kubernetes.io/docs/reference/access-authn-authz/kubelet-authn-authz/#kubelet-authentication
3. https://kubernetes.io/docs/reference/config-api/kubelet-config.v1beta1/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.3 Disable Dormant Accounts Delete or disable any dormant accounts after a period of 45 days of inactivity, where supported.	●	●	●
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.	●	●	●

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

Profile Applicability:

- Level 1

Description:

Do not allow all requests. Enable explicit authorization.

Rationale:

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

Impact:

Unauthorized requests will be denied.

Audit:

Audit Method 1:

Kubelets can accept configuration via a configuration file and in some cases via command line arguments. It is important to note that parameters provided as command line arguments will override their counterpart parameters in the configuration file (see `--config` details in the Kubelet CLI Reference for more info, where you can also find out which configuration parameters can be supplied as a command line argument).

With this in mind, it is important to check for the existence of command line arguments as well as configuration file entries when auditing Kubelet configuration.

Firstly, SSH to each node and execute the following command to find the Kubelet process:

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

The output of the above command provides details of the active Kubelet process, from which we can see the command line arguments provided to the process. Also note the location of the configuration file, provided with the `--config` argument, as this will be needed to verify configuration. The file can be viewed with a command such as `more` or `less`, like so:

```
sudo less /path/to/kubelet-config.json
```

Verify that Webhook Authentication is enabled. This may be enabled as a command line argument to the kubelet service with `--authentication-token-webhook` or in the kubelet configuration file via `"authentication": { "webhook": { "enabled": true } }`.

Verify that the Authorization Mode is set to `WebHook`. This may be set as a command line argument to the kubelet service with `--authorization-mode=Webhook` or in the configuration file via `"authorization": { "mode": "Webhook }`.

Audit Method 2:

It is also possible to review the running configuration of a Kubelet via the /configz endpoint of the Kubernetes API. This can be achieved using `kubectl` to proxy your requests to the API.

Discover all nodes in your cluster by running the following command:

```
kubectl get nodes
```

Next, initiate a proxy with kubectl on a local port of your choice. In this example we will use 8080:

```
kubectl proxy --port=8080
```

With this running, in a separate terminal run the following command for each node:

```
export NODE_NAME=my-node-name
curl http://localhost:8080/api/v1/nodes/${NODE_NAME}/proxy/configz
```

The curl command will return the API response which will be a JSON formatted string representing the Kubelet configuration.

Verify that Webhook Authentication is enabled with `"authentication": { "webhook": { "enabled": true } }` in the API response.

Verify that the Authorization Mode is set to `WebHook` with `"authorization": { "mode": "Webhook }` in the API response.

Remediation:

Remediation Method 1:

If configuring via the Kubelet config file, you first need to locate the file.

To do this, SSH to each node and execute the following command to find the kubelet process:

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

The output of the above command provides details of the active kubelet process, from which we can see the location of the configuration file provided to the kubelet service with the `--config` argument. The file can be viewed with a command such as `more` or `less`, like so:

```
sudo less /path/to/kubelet-config.json
```

Enable Webhook Authentication by setting the following parameter:

```
"authentication": { "webhook": { "enabled": true } }
```

Next, set the Authorization Mode to `Webhook` by setting the following parameter:

```
"authorization": { "mode": "Webhook }
```

Finer detail of the `authentication` and `authorization` fields can be found in the Kubelet Configuration documentation.

Remediation Method 2:

If using executable arguments, edit the kubelet service file on each worker node and ensure the below parameters are part of the `KUBELET_ARGS` variable string.

For systems using `systemd`, such as the Amazon EKS Optimised Amazon Linux or Bottlerocket AMIs, then this file can be found at `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf`. Otherwise, you may need to look up documentation for your chosen operating system to determine which service manager is configured:

```
--authentication-token-webhook
--authorization-mode=Webhook
```

For Both Remediation Steps:

Based on your system, restart the `kubelet` service and check the service status.

The following example is for operating systems using `systemd`, such as the Amazon EKS Optimised Amazon Linux or Bottlerocket AMIs, and invokes the `systemctl` command. If `systemctl` is not available then you will need to look up documentation for your chosen operating system to determine which service manager is configured:

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

Default Value:

See the EKS documentation for the default value.

References:

1. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
2. https://kubernetes.io/docs/reference/access-authn-authz/kubelet-authn-authz/#kubelet-authentication
3. https://kubernetes.io/docs/reference/config-api/kubelet-config.v1beta1/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.2 Change Default Passwords Before deploying any new asset, change all default passwords to have values consistent with administrative level accounts.	●	●	●

3.2.3 Ensure that a Client CA File is Configured (Automated)

Profile Applicability:

- Level 1

Description:

Enable Kubelet authentication using certificates.

Rationale:

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

Impact:

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

Audit:

Audit Method 1:

Kubelets can accept configuration via a configuration file and in some cases via command line arguments. It is important to note that parameters provided as command line arguments will override their counterpart parameters in the configuration file (see `--config` details in the Kubelet CLI Reference for more info, where you can also find out which configuration parameters can be supplied as a command line argument).

With this in mind, it is important to check for the existence of command line arguments as well as configuration file entries when auditing Kubelet configuration.

Firstly, SSH to each node and execute the following command to find the Kubelet process:

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

The output of the above command provides details of the active Kubelet process, from which we can see the command line arguments provided to the process. Also note the location of the configuration file, provided with the `--config` argument, as this will be needed to verify configuration. The file can be viewed with a command such as `more` or `less`, like so:

```
sudo less /path/to/kubelet-config.json
```

Verify that a client certificate authority file is configured. This may be configured using a command line argument to the kubelet service with `--client-ca-file` or in the kubelet configuration file via `"authentication": { "x509": {"clientCAFile": <path/to/client-ca-file> } }`.

Audit Method 2:

It is also possible to review the running configuration of a Kubelet via the /configz endpoint of the Kubernetes API. This can be achieved using `kubectl` to proxy your requests to the API.

Discover all nodes in your cluster by running the following command:

```
kubectl get nodes
```

Next, initiate a proxy with kubectl on a local port of your choice. In this example we will use 8080:

```
kubectl proxy --port=8080
```

With this running, in a separate terminal run the following command for each node:

```
export NODE_NAME=my-node-name
curl http://localhost:8080/api/v1/nodes/${NODE_NAME}/proxy/configz
```

The curl command will return the API response which will be a JSON formatted string representing the Kubelet configuration.

Verify that a client certificate authority file is configured with `"authentication": { "x509": {"clientCAFile": <path/to/client-ca-file> } }"` in the API response.

Remediation:

Remediation Method 1:

If configuring via the Kubelet config file, you first need to locate the file.

To do this, SSH to each node and execute the following command to find the kubelet process:

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

The output of the above command provides details of the active kubelet process, from which we can see the location of the configuration file provided to the kubelet service with the `--config` argument. The file can be viewed with a command such as `more` or `less`, like so:

```
sudo less /path/to/kubelet-config.json
```

Configure the client certificate authority file by setting the following parameter appropriately:

```
"authentication": { "x509": {"clientCAFile": <path/to/client-ca-file> } }"
```

Remediation Method 2:

If using executable arguments, edit the kubelet service file on each worker node and ensure the below parameters are part of the `KUBELET_ARGS` variable string.

For systems using `systemd`, such as the Amazon EKS Optimised Amazon Linux or Bottlerocket AMIs, then this file can be found at `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf`. Otherwise, you may need to look up documentation for your chosen operating system to determine which service manager is configured:

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

For Both Remediation Steps:

Based on your system, restart the `kubelet` service and check the service status.

The following example is for operating systems using `systemd`, such as the Amazon EKS Optimised Amazon Linux or Bottlerocket AMIs, and invokes the `systemctl` command. If `systemctl` is not available then you will need to look up documentation for your chosen operating system to determine which service manager is configured:

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

Default Value:

See the EKS documentation for the default value.

References:

1. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
2. https://kubernetes.io/docs/reference/access-authn-authz/kubelet-authn-authz/#kubelet-authentication
3. https://kubernetes.io/docs/reference/config-api/kubelet-config.v1beta1/

CIS Controls:

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

3.2.4 Ensure that the --read-only-port is disabled (Automated)

Profile Applicability:

- Level 1

Description:

Disable the read-only port.

Rationale:

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

Impact:

Removal of the read-only port will require that any service which made use of it will need to be re-configured to use the main Kubelet API.

Audit:

If using a Kubelet configuration file, check that there is an entry for `authentication: anonymous: enabled` set to `0`.

First, SSH to the relevant node:

Run the following command on each node to find the appropriate Kubelet config file:

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

The output of the above command should return something similar to `--config /etc/kubernetes/kubelet/kubelet-config.json` which is the location of the Kubelet config file.

Open the Kubelet config file:

```
cat /etc/kubernetes/kubelet/kubelet-config.json
```

Verify that the `--read-only-port` argument exists and is set to `0`.

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

Remediation:

If modifying the Kubelet config file, edit the kubelet-config.json file `/etc/kubernetes/kubelet/kubelet-config.json` and set the below parameter to 0

```
"readOnlyPort": 0
```

If using executable arguments, edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` on each worker node and add the below parameter at the end of the `KUBELET_ARGS` variable string.

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

For each remediation: Based on your system, restart the `kubelet` service and check status

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

Default Value:

See the Amazon EKS documentation for the default value.

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Do not disable timeouts on streaming connections.

Rationale:

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

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

Impact:

Long-lived connections could be interrupted.

Audit:

Audit Method 1:

First, SSH to the relevant node:

Run the following command on each node to find the running kubelet process:

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

If the command line for the process includes the argument `streaming-connection-idle-timeout` verify that it is not set to 0.

If the `streaming-connection-idle-timeout` argument is not present in the output of the above command, refer instead to the `config` argument that specifies the location of the Kubelet config file e.g. `--config /etc/kubernetes/kubelet/kubelet-config.json`.

Open the Kubelet config file:

```
cat /etc/kubernetes/kubelet/kubelet-config.json
```

Verify that the `streamingConnectionIdleTimeout` argument is not set to `0`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `"streamingConnectionIdleTimeout":"4h0m0s"` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number"` `NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=ip-192.168.31.226.ec2.internal (example node name from
"kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet config file, edit the kubelet-config.json file `/etc/kubernetes/kubelet/kubelet-config.json` and set the below parameter to a non-zero value in the format of #h#m#s

```
"streamingConnectionIdleTimeout": "4h0m0s"
```

You should ensure that the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` does not specify a `--streaming-connection-idle-timeout` argument because it would override the Kubelet config file.

Remediation Method 2:

If using executable arguments, edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` on each worker node and add the below parameter at the end of the `KUBELET_ARGS` variable string.

```
--streaming-connection-idle-timeout=4h0m0s
```

Remediation Method 3:

If using the api configz endpoint consider searching for the status of `"streamingConnectionIdleTimeout":` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in [Reconfigure a Node's Kubelet in a Live Cluster](), and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=ip-192.168.31.226.ec2.internal (example node name from
"kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all three remediations: Based on your system, restart the `kubelet` service and check status

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

Default Value:

See the EKS documentation for the default value.

References:

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

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Allow Kubelet to manage iptables.

Rationale:

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

Impact:

Kubelet would manage the iptables on the system and keep it in sync. If you are using any other iptables management solution, then there might be some conflicts.

Audit:

Audit Method 1:

First, SSH to each node:

Run the following command on each node to find the Kubelet process:

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

If the output of the above command includes the argument `--make-iptables-util-chains` then verify it is set to true.

If the `--make-iptables-util-chains` argument does not exist, and there is a Kubelet config file specified by `--config`, verify that the file does not set `makeIPTablesUtilChains` to `false`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `authentication... "makeIPTablesUtilChains.:true` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number"` `NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=ip-192.168.31.226.ec2.internal (example node name from
"kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet config file, edit the kubelet-config.json file `/etc/kubernetes/kubelet/kubelet-config.json` and set the below parameter to true

```
"makeIPTablesUtilChains": true
```

Ensure that `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` does not set the `--make-iptables-util-chains` argument because that would override your Kubelet config file.

Remediation Method 2:

If using executable arguments, edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` on each worker node and add the below parameter at the end of the `KUBELET_ARGS` variable string.

```
--make-iptables-util-chains:true
```

Remediation Method 3:

If using the api configz endpoint consider searching for the status of `"makeIPTablesUtilChains.: true` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=ip-192.168.31.226.ec2.internal (example node name from
"kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all three remediations: Based on your system, restart the `kubelet` service and check status

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

Default Value:

See the Amazon EKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.3 Securely Manage Network Infrastructure Securely manage network infrastructure. Example implementations include version-controlled-infrastructure-as-code, and the use of secure network protocols, such as SSH and HTTPS.		●	●
v7	11.1 Maintain Standard Security Configurations for Network Devices Maintain standard, documented security configuration standards for all authorized network devices.		●	●

3.2.7 Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture (Automated)

Profile Applicability:

- Level 1

Description:

Security relevant information should be captured. The eventRecordQPS on the Kubelet configuration can be used to limit the rate at which events are gathered and sets the maximum event creations per second. Setting this too low could result in relevant events not being logged, however the unlimited setting of `0` could result in a denial of service on the kubelet.

Rationale:

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

Impact:

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

Audit:

Run the following command on each node:

```
sudo grep "eventRecordQPS" /etc/systemd/system/kubelet.service.d/10-
kubeadm.conf
```

Review the value set for the argument and determine whether this has been set to an appropriate level for the cluster.

If the argument does not exist, check that there is a Kubelet config file specified by `--config` and review the value in this location.

Remediation:

If using a Kubelet config file, edit the file to set `eventRecordQPS:` to an appropriate level.

If using command line arguments, edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and set the below parameter in `KUBELET_SYSTEM_PODS_ARGS` variable.

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

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

Default Value:

See the Amazon EKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/apis/kubeletconfig/v1beta1/types.go
3. https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.	●	●	●
v8	8.5 Collect Detailed Audit Logs Configure detailed audit logging for enterprise assets containing sensitive data. Include event source, date, username, timestamp, source addresses, destination addresses, and other useful elements that could assist in a forensic investigation.		●	●
v7	6.2 Activate audit logging Ensure that local logging has been enabled on all systems and networking devices.	●	●	●
v7	6.3 Enable Detailed Logging Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.		●	●

3.2.8 Ensure that the --rotate-certificates argument is not present or is set to true (Automated)

Profile Applicability:

- Level 1

Description:

Enable kubelet client certificate rotation.

Rationale:

The `--rotate-certificates` setting causes the kubelet to rotate its client certificates by creating new CSRs as its existing credentials expire. This automated periodic rotation ensures that the there is no downtime due to expired certificates and thus addressing availability in the CIA (Confidentiality, Integrity, and Availability) 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 implement rotation yourself.

Note: This feature also requires the `RotateKubeletClientCertificate` feature gate to be enabled.

Impact:

None

Audit:

Audit Method 1:

SSH to each node and run the following command to find the Kubelet process:

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

If the output of the command above includes the `--RotateCertificate` executable argument, verify that it is set to true. If the output of the command above does not include the `--RotateCertificate` executable argument then check the Kubelet config file. The output of the above command should return something similar to `--config /etc/kubernetes/kubelet/kubelet-config.json` which is the location of the Kubelet config file.

Open the Kubelet config file:

```
cat /etc/kubernetes/kubelet/kubelet-config.json
```

Verify that the `RotateCertificate` argument is not present, or is set to `true`.

Remediation:

Remediation Method 1:

If modifying the Kubelet config file, edit the kubelet-config.json file `/etc/kubernetes/kubelet/kubelet-config.json` and set the below parameter to true

```
"RotateCertificate":true
```

Additionally, ensure that the kubelet service file /etc/systemd/system/kubelet.service.d/10-kubelet-args.conf does not set the --RotateCertificate executable argument to false because this would override the Kubelet config file.

Remediation Method 2:

If using executable arguments, edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` on each worker node and add the below parameter at the end of the `KUBELET_ARGS` variable string.

```
--RotateCertificate=true
```

Default Value:

See the Amazon EKS documentation for the default value.

References:

1. https://github.com/kubernetes/kubernetes/pull/41912
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-tls-bootstrapping/#kubelet-configuration
3. https://kubernetes.io/docs/imported/release/notes/
4. https://kubernetes.io/docs/reference/command-line-tools-reference/feature-gates/
5. https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Enable kubelet server certificate rotation.

Rationale:

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

Impact:

None

Audit:

Audit Method 1:

First, SSH to each node:

Run the following command on each node to find the Kubelet process:

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

If the output of the command above includes the `--rotate-kubelet-server-certificate` executable argument verify that it is set to true.

If the process does not have the `--rotate-kubelet-server-certificate` executable argument then check the Kubelet config file. The output of the above command should return something similar to `--config /etc/kubernetes/kubelet/kubelet-config.json` which is the location of the Kubelet config file.

Open the Kubelet config file:

```
cat /etc/kubernetes/kubelet/kubelet-config.json
```

Verify that `RotateKubeletServerCertificate` argument exists in the `featureGates` section and is set to `true`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `"RotateKubeletServerCertificate":true` by extracting the live configuration from the nodes running kubelet.

Set the local proxy port and the following variables and provide proxy port number and node name; `HOSTNAME_PORT="localhost-and-port-number"` `NODE_NAME="The-Name-Of-Node-To-Extract-Configuration" from the output of "kubectl get nodes"`

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=ip-192.168.31.226.ec2.internal (example node name from
"kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

Remediation:

Remediation Method 1:

If modifying the Kubelet config file, edit the kubelet-config.json file `/etc/kubernetes/kubelet/kubelet-config.json` and set the below parameter to true

```
"featureGates": {
  "RotateKubeletServerCertificate":true
},
```

Additionally, ensure that the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` does not set the `--rotate-kubelet-server-certificate` executable argument to false because this would override the Kubelet config file.

Remediation Method 2:

If using executable arguments, edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` on each worker node and add the below parameter at the end of the `KUBELET_ARGS` variable string.

```
--rotate-kubelet-server-certificate=true
```

Remediation Method 3:

If using the api configz endpoint consider searching for the status of `"RotateKubeletServerCertificate":` by extracting the live configuration from the nodes running kubelet.

**See detailed step-by-step configmap procedures in Reconfigure a Node's Kubelet in a Live Cluster, and then rerun the curl statement from audit process to check for kubelet configuration changes

```
kubectl proxy --port=8001 &

export HOSTNAME_PORT=localhost:8001 (example host and port number)
export NODE_NAME=ip-192.168.31.226.ec2.internal (example node name from
"kubectl get nodes")

curl -sSL "http://${HOSTNAME_PORT}/api/v1/nodes/${NODE_NAME}/proxy/configz"
```

For all three remediation methods: Restart the `kubelet` service and check status. The example below is for when using systemctl to manage services:

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

Default Value:

See the Amazon EKS documentation for the default value.

References:

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

CIS Controls:

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

4 Policies

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

4.1 RBAC and Service Accounts

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

Profile Applicability:

- Level 1

Description:

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

Rationale:

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

Impact:

Care should be taken before removing any `ClusterRoleBindings` from the environment to ensure they were not required for operation of the cluster. Specifically, modifications should not be made to `ClusterRoleBindings` with the `system:` prefix as they are required for the operation of system components.

Audit:

Obtain a list of the principals who have access to the `cluster-admin` role by reviewing the `clusterrolebinding` output for each role binding that has access to the `cluster-admin` role.

kubectl get clusterrolebindings -o=custom-columns=NAME:.metadata.name,ROLE:.roleRef.name,SUBJECT:.subjects[*].name

Review each principal listed and ensure that `cluster-admin` privilege is required for it.

Remediation:

Identify all ClusterRoleBindings to the cluster-admin role. Check if they are used and if they need this role or if they could use a role with fewer privileges.

Where possible, first bind users to a lower privileged role and then remove the ClusterRoleBinding to the cluster-admin role :

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

Default Value:

By default a single `clusterrolebinding` called `cluster-admin` is provided with the `system:masters` group as its principal.

References:

1. https://kubernetes.io/docs/admin/authorization/rbac/#user-facing-roles

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

4.1.2 Minimize access to secrets (Manual)

Profile Applicability:

- Level 1

Description:

The Kubernetes API stores secrets, which may be service account tokens for the Kubernetes API or credentials used by workloads in the cluster. Access to these secrets should be restricted to the smallest possible group of users to reduce the risk of privilege escalation.

Rationale:

Inappropriate access to secrets stored within the Kubernetes cluster can allow for an attacker to gain additional access to the Kubernetes cluster or external resources whose credentials are stored as secrets.

Impact:

Care should be taken not to remove access to secrets to system components which require this for their operation

Audit:

Review the users who have `get`, `list` or `watch` access to `secrets` objects in the Kubernetes API.

Remediation:

Where possible, remove `get`, `list` and `watch` access to `secret` objects in the cluster.

Default Value:

By default, the following list of principals have `get` privileges on `secret` objects

```
CLUSTERROLEBINDING                                     SUBJECT
TYPE            SA-NAMESPACE
cluster-admin                                          system:masters
Group
system:controller:clusterrole-aggregation-controller  clusterrole-
aggregation-controller  ServiceAccount  kube-system
system:controller:expand-controller                    expand-controller
ServiceAccount  kube-system
system:controller:generic-garbage-collector            generic-garbage-
collector           ServiceAccount  kube-system
system:controller:namespace-controller                 namespace-controller
ServiceAccount  kube-system
system:controller:persistent-volume-binder             persistent-volume-
binder            ServiceAccount  kube-system
system:kube-controller-manager                         system:kube-controller-
manager      User
```

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 Establish and Maintain a Secure Configuration Process Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.	●	●	●
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

4.1.3 Minimize wildcard use in Roles and ClusterRoles (Manual)

Profile Applicability:

- Level 1

Description:

Kubernetes Roles and ClusterRoles provide access to resources based on sets of objects and actions that can be taken on those objects. It is possible to set either of these to be the wildcard "*" which matches all items.

Use of wildcards is not optimal from a security perspective as it may allow for inadvertent access to be granted when new resources are added to the Kubernetes API either as CRDs or in later versions of the product.

Rationale:

The principle of least privilege recommends that users are provided only the access required for their role and nothing more. The use of wildcard rights grants is likely to provide excessive rights to the Kubernetes API.

Audit:

Retrieve the roles defined across each namespaces in the cluster and review for wildcards

```
kubectl get roles --all-namespaces -o yaml
```

Retrieve the cluster roles defined in the cluster and review for wildcards

```
kubectl get clusterroles -o yaml
```

Remediation:

Where possible replace any use of wildcards in clusterroles and roles with specific objects or actions.

CIS Controls:

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

4.1.4 Minimize access to create pods (Manual)

Profile Applicability:

- Level 1

Description:

The ability to create pods in a namespace can provide a number of opportunities for privilege escalation, such as assigning privileged service accounts to these pods or mounting hostPaths with access to sensitive data (unless Pod Security Policies are implemented to restrict this access)

As such, access to create new pods should be restricted to the smallest possible group of users.

Rationale:

The ability to create pods in a cluster opens up possibilities for privilege escalation and should be restricted, where possible.

Impact:

Care should be taken not to remove access to pods to system components which require this for their operation

Audit:

Review the users who have create access to pod objects in the Kubernetes API.

Remediation:

Where possible, remove `create` access to `pod` objects in the cluster.

Default Value:

By default, the following list of principals have `create` privileges on `pod` objects

```
CLUSTERROLEBINDING                                     SUBJECT
TYPE            SA-NAMESPACE
cluster-admin                                          system:masters
Group
system:controller:clusterrole-aggregation-controller  clusterrole-
aggregation-controller  ServiceAccount  kube-system
system:controller:daemon-set-controller                daemon-set-controller
ServiceAccount  kube-system
system:controller:job-controller                       job-controller
ServiceAccount  kube-system
system:controller:persistent-volume-binder             persistent-volume-
binder            ServiceAccount  kube-system
system:controller:replicaset-controller                replicaset-controller
ServiceAccount  kube-system
system:controller:replication-controller               replication-controller
ServiceAccount  kube-system
system:controller:statefulset-controller               statefulset-controller
ServiceAccount  kube-system
```

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	2.7 Allowlist Authorized Scripts Use technical controls, such as digital signatures and version control, to ensure that only authorized scripts, such as specific .ps1, .py, etc., files, are allowed to execute. Block unauthorized scripts from executing. Reassess bi-annually, or more frequently.			●
v7	4.7 Limit Access to Script Tools Limit access to scripting tools (such as Microsoft PowerShell and Python) to only administrative or development users with the need to access those capabilities.		●	●

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

Profile Applicability:

- Level 1

Description:

The `default` service account should not be used to ensure that rights granted to applications can be more easily audited and reviewed.

Rationale:

Kubernetes provides a `default` service account which is used by cluster workloads where no specific service account is assigned to the pod.

Where access to the Kubernetes API from a pod is required, a specific service account should be created for that pod, and rights granted to that service account.

The default service account should be configured such that it does not provide a service account token and does not have any explicit rights assignments.

Impact:

All workloads which require access to the Kubernetes API will require an explicit service account to be created.

Audit:

For each namespace in the cluster, review the rights assigned to the default service account and ensure that it has no roles or cluster roles bound to it apart from the defaults.

Additionally ensure that the `automountServiceAccountToken: false` setting is in place for each default service account.

Remediation:

Create explicit service accounts wherever a Kubernetes workload requires specific access to the Kubernetes API server.

Modify the configuration of each default service account to include this value

```
automountServiceAccountToken: false
```

Automatic remediation for the default account:

```
kubectl patch serviceaccount default -p
$'automountServiceAccountToken: false'
```

Default Value:

By default the `default` service account allows for its service account token to be mounted in pods in its namespace.

References:

1. https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/
2. https://aws.github.io/aws-eks-best-practices/iam/#disable-auto-mounting-of-service-account-tokens

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.3 Disable Dormant Accounts Delete or disable any dormant accounts after a period of 45 days of inactivity, where supported.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●
v7	16.9 Disable Dormant Accounts Automatically disable dormant accounts after a set period of inactivity.	●	●	●

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

Profile Applicability:

- Level 1

Description:

Service accounts tokens should not be mounted in pods except where the workload running in the pod explicitly needs to communicate with the API server

Rationale:

Mounting service account tokens inside pods can provide an avenue for privilege escalation attacks where an attacker is able to compromise a single pod in the cluster.

Avoiding mounting these tokens removes this attack avenue.

Impact:

Pods mounted without service account tokens will not be able to communicate with the API server, except where the resource is available to unauthenticated principals.

Audit:

Review pod and service account objects in the cluster and ensure that the option below is set, unless the resource explicitly requires this access.

```
automountServiceAccountToken: false
```

Remediation:

Regularly review pod and service account objects in the cluster to ensure that the `automountServiceAccountToken` setting is `false` for pods and accounts that do not explicitly require API server access.

Default Value:

By default, all pods get a service account token mounted in them.

References:

1. https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	14.7 Enforce Access Control to Data through Automated Tools Use an automated tool, such as host-based Data Loss Prevention, to enforce access controls to data even when data is copied off a system.			●

4.1.7 Cluster Access Manager API to streamline and enhance the management of access controls within EKS clusters (Automated)

Profile Applicability:

- Level 1

Description:

Amazon EKS has introduced the Cluster Access Manager API to streamline and enhance the management of access controls within EKS clusters. This new approach is now the recommended method over the traditional `aws-auth` ConfigMap for managing Role-Based Access Control (RBAC) and Service Accounts.

Key Advantages of Using the Cluster Access Manager API:

1. **Simplified Access Management:** The Cluster Access Manager API allows administrators to manage access directly through the Amazon EKS API, eliminating the need to modify the aws-auth ConfigMap manually. This reduces operational overhead and minimizes the risk of misconfigurations.
2. **Enhanced Security Controls:** With this API, administrators can assign predefined AWS-managed Kubernetes permissions, known as "access policies," to IAM principals. This provides a more secure and auditable way to manage permissions compared to manual ConfigMap edits.
3. **Improved Visibility and Auditing:** The API offers better visibility into cluster access configurations, facilitating easier auditing and compliance checks. Administrators can list and describe access entries and policies directly through the EKS API.

Rationale:

The compelling rationale for using the Cluster Access Manager API instead of the traditional aws-auth ConfigMap in Amazon EKS revolves around security, scalability, operational efficiency, and simplified management.

1. Increased Security and Reduced Risk

- Direct Management via API: The Cluster Access Manager API enables you to manage RBAC and IAM permissions directly through the EKS API rather than editing a ConfigMap. This eliminates the risk of inadvertent errors when manually modifying the `aws-auth` ConfigMap.
- Immutable Access Entries: The API ensures that once access entries are defined, they are tightly controlled, reducing the risk of accidental overwrites or misconfigurations that can happen when editing YAML files.
- Fine-Grained Access Control: By leveraging the new API, you can define access policies at a more granular level than the previous method. This ensures that only the necessary permissions are granted, minimizing the attack surface.

2. Operational Efficiency and Scalability

- Scalability: Managing access control through the aws-auth ConfigMap becomes increasingly challenging as the number of users and services grows. The new API scales better by allowing access management through standard AWS Identity and Access Management (IAM) tools.
- Reduced Operational Overhead: The API simplifies the management of access controls by removing the need for manual updates to the ConfigMap, reducing the risk of human error, and automating access provisioning through Infrastructure as Code (IaC) tools like Terraform or CloudFormation.

3. Improved Visibility, Auditing, and Compliance

- Auditable and Traceable Changes: The Cluster Access Manager API integrates with AWS CloudTrail, allowing you to track who made changes to access configurations. This level of visibility is critical for organizations that need to adhere to compliance frameworks like SOC 2, GDPR, or HIPAA.
- Centralized Management: Unlike the `aws-auth` ConfigMap, which is managed at the Kubernetes level, the new API leverages AWS IAM's centralized management and auditing capabilities, providing a unified view of access controls across your AWS environment.

4. Faster and Safer Access Provisioning

- No More Cluster Downtime: Errors in the aws-auth ConfigMap can accidentally lock out users or admins from the cluster, requiring complex recovery processes. The API-based approach is more resilient, reducing the risk of misconfigurations causing downtime.
- Immediate Effect: Changes made via the API take effect immediately, whereas updates to the aws-auth ConfigMap may require a delay or even restarting components in some cases.

5. Future-Proofing and Alignment with AWS Best Practices

- Native Support in Kubernetes Versions: Starting from Kubernetes 1.23, the Cluster Access Manager API is fully supported and designed to replace the aws-auth ConfigMap method. This aligns with AWS's roadmap and best practices for EKS, ensuring your infrastructure remains compatible with future updates.
- Modern Approach for Pod Identity: When combined with IAM Roles for Service Accounts (IRSA) or the new Pod Identity feature, the API supports a more dynamic and secure model for assigning permissions to pods, making it easier to implement least-privilege access.

Impact:

The shift to using the **Cluster Access Manager API** instead of the aws-auth ConfigMap impacts EKS RBAC and Service Accounts by simplifying access control management, reducing the risk of misconfigurations, and enhancing security. It allows for more granular, direct management of IAM permissions and Kubernetes roles, eliminating manual ConfigMap edits and reducing operational overhead. For Service Accounts, it better integrates with existing mechanisms like IAM Roles for Service Accounts (IRSA) for secure pod access to AWS resources, making it easier to enforce least-privilege principles. This transition improves scalability, auditing, and compliance, while providing a future-proof solution aligned with AWS's Kubernetes identity management roadmap.

Audit:

To check if the Cluster Access Manager API is active on your Amazon EKS cluster, you can use the following AWS CLI command:

```
aws eks describe-cluster --name $CLUSTER_NAME --query
"cluster.accessConfig" --output json
```

Replace $CLUSTER_NAME with the name of your EKS cluster.

The command queries the `cluster.accessConfig` property, which indicates the authentication mode of the cluster.

Possible Outputs: If the output shows `"authenticationmode": "API" or "authenticationmode": "API_AND_CONFIG_MAP"`, it means the Cluster Access Manager API is enabled.

If it only shows `"authenticationmode": "CONFIG_MAP"`, then the cluster is still using the traditional `aws-auth` ConfigMap approach.

Remediation:

Log in to the AWS Management Console.

Navigate to Amazon EKS and select your EKS cluster.

Go to the Access tab and click on "Manage Access" in the "Access Configuration section".

Under Cluster Authentication Mode for Cluster Access settings.

- Click `EKS API` to change `cluster will source authenticated IAM principals only from EKS access entry APIs`.
- Click `ConfigMap` to change `cluster will source authenticated IAM principals only from the aws-auth ConfigMap`.
- Note: `EKS API and ConfigMap` must be selected during Cluster creation and cannot be changed once the Cluster is provisioned.

Default Value:

`EKS API` is selected by default during EKS Cluster creation but can be changed during initial configuration

References:

1. https://aws.amazon.com/blogs/containers/a-deep-dive-into-simplified-amazon-eks-access-management-controls/
2. https://www.eksworkshop.com/docs/security/cluster-access-management/understanding

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	14.7 Enforce Access Control to Data through Automated Tools Use an automated tool, such as host-based Data Loss Prevention, to enforce access controls to data even when data is copied off a system.			●

4.1.8 Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster (Manual)

Profile Applicability:

- Level 1

Description:

Cluster roles and roles with the impersonate, bind or escalate permissions should not be granted unless strictly required. Each of these permissions allow a particular subject to escalate their privileges beyond those explicitly granted by cluster administrators

Rationale:

The impersonate privilege allows a subject to impersonate other users gaining their rights to the cluster. The bind privilege allows the subject to add a binding to a cluster role or role which escalates their effective permissions in the cluster. The escalate privilege allows a subject to modify cluster roles to which they are bound, increasing their rights to that level.

Each of these permissions has the potential to allow for privilege escalation to cluster-admin level.

Impact:

There are some cases where these permissions are required for cluster service operation, and care should be taken before removing these permissions from system service accounts.

Audit:

Review the users who have access to cluster roles or roles which provide the impersonate, bind or escalate privileges.

Remediation:

Where possible, remove the impersonate, bind and escalate rights from subjects.

Default Value:

In a default kubeadm cluster, the system:masters group and clusterrole-aggregation-controller service account have access to the escalate privilege. The system:masters group also has access to bind and impersonate.

References:

1. https://www.impidio.com/blog/kubernetes-rbac-security-pitfalls
2. https://raesene.github.io/blog/2020/12/12/Escalating_Away/
3. https://raesene.github.io/blog/2021/01/16/Getting-Into-A-Bind-with-Kubernetes/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	14.7 Enforce Access Control to Data through Automated Tools Use an automated tool, such as host-based Data Loss Prevention, to enforce access controls to data even when data is copied off a system.			●

4.1.9 Minimize access to create persistent volumes (Manual)

Profile Applicability:

- Level 1

Description:

The ability to create persistent volumes in a cluster can provide an opportunity for privilege escalation, via the creation of `hostPath` volumes. As persistent volumes are not covered by Pod Security Admission, a user with access to create persistent volumes may be able to get access to sensitive files from the underlying host even where restrictive Pod Security Admission policies are in place.

Rationale:

The ability to create persistent volumes in a cluster opens up possibilities for privilege escalation and should be restricted, where possible.

Audit:

Review the users who have create access to `PersistentVolume` objects in the Kubernetes API.

Remediation:

Where possible, remove `create` access to `PersistentVolume` objects in the cluster.

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#persistent-volume-creation

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●

4.1.10 Minimize access to the proxy sub-resource of nodes (Manual)

Profile Applicability:

- Level 1

Description:

Users with access to the `Proxy` sub-resource of `Node` objects automatically have permissions to use the kubelet API, which may allow for privilege escalation or bypass cluster security controls such as audit logs.

The kubelet provides an API which includes rights to execute commands in any container running on the node. Access to this API is covered by permissions to the main Kubernetes API via the `node` object. The proxy sub-resource specifically allows wide ranging access to the kubelet API.

Direct access to the kubelet API bypasses controls like audit logging (there is no audit log of kubelet API access) and admission control.

Rationale:

The ability to use the `proxy` sub-resource of `node` objects opens up possibilities for privilege escalation and should be restricted, where possible.

Impact:

Review the users who have access to the `proxy` sub-resource of `node` objects in the Kubernetes API.

Audit:

Where possible, remove access to the `proxy` sub-resource of `node` objects.

Remediation:

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#access-to-proxy-subresource-of-nodes
2. https://kubernetes.io/docs/reference/access-authn-authz/kubelet-authn-authz/#kubelet-authorization

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 <u>Define and Maintain Role-Based Access Control</u> Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●

4.1.11 Minimize access to webhook configuration objects (Manual)

Profile Applicability:

- Level 1

Description:

Users with rights to create/modify/delete `validatingwebhookconfigurations` or `mutatingwebhookconfigurations` can control webhooks that can read any object admitted to the cluster, and in the case of mutating webhooks, also mutate admitted objects. This could allow for privilege escalation or disruption of the operation of the cluster.

Rationale:

The ability to manage webhook configuration should be limited

Audit:

Review the users who have access to `validatingwebhookconfigurations` or `mutatingwebhookconfigurations` objects in the Kubernetes API.

Remediation:

Where possible, remove access to the `validatingwebhookconfigurations` or `mutatingwebhookconfigurations` objects

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#control-admission-webhooks

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 <u>Define and Maintain Role-Based Access Control</u> Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●

4.1.12 Minimize access to the service account token creation (Manual)

Profile Applicability:

- Level 1

Description:

Users with rights to create new service account tokens at a cluster level, can create long-lived privileged credentials in the cluster. This could allow for privilege escalation and persistent access to the cluster, even if the users account has been revoked.

Rationale:

The ability to create service account tokens should be limited.

Audit:

Review the users who have access to create the `token` sub-resource of `serviceaccount` objects in the Kubernetes API.

Remediation:

Where possible, remove access to the `token` sub-resource of `serviceaccount` objects.

References:

1. https://kubernetes.io/docs/concepts/security/rbac-good-practices/#token-request

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●

4.2 Pod Security Standards

Pod Security Standards (PSS) are recommendations for securing deployed workloads to reduce the risks of container breakout. There are a number of ways if implementing PSS, including the built-in Pod Security Admission controller, or external policy control systems which integrate with Kubernetes via validating and mutating webhooks.

The previous feature described in this document, pod security policy (preview), was deprecated with version 1.21, and removed as of version 1.25. After pod security policy (preview) is deprecated, you must disable the feature on any existing clusters using the deprecated feature to perform future cluster upgrades and stay within Azure support.

4.2.1 Minimize the admission of privileged containers (Manual)

Profile Applicability:

- Level 1

Description:

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

Rationale:

Privileged containers have access to all Linux Kernel capabilities and devices. A container running with full privileges can do almost everything that the host can do. This flag exists to allow special use-cases, like manipulating the network stack and accessing devices.

There should be at least one admission control policy defined which does not permit privileged containers.

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

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of privileged containers.

Since manually searching through each pod's configuration might be tedious, especially in environments with many pods, you can use a more automated approach with grep or other command-line tools.

Here's an example of how you might approach this with a combination of kubectl, grep, and shell scripting for a more automated solution:

```
kubectl get pods --all-namespaces -o json | jq -r '.items[] |
select(.spec.containers[].securityContext.privileged == true) |
.metadata.name'
```

OR

```
kubectl get pods --all-namespaces -o json | jq '.items[] |
select(.metadata.namespace != "kube-system" and
.spec.containers[]?.securityContext?.privileged == true) | {pod:
.metadata.name, namespace: .metadata.namespace, container:
.spec.containers[].name}'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command uses `jq`, a command-line JSON processor, to parse the JSON output from kubectl get pods and filter out pods where any container has the securityContext.privileged flag set to true. Please note that you might need to adjust the command depending on your specific requirements and the structure of your pod specifications.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of privileged containers.

To enable PSA for a namespace in your cluster, set the pod-security.kubernetes.io/enforce label with the policy value you want to enforce.

`kubectl label --overwrite ns NAMESPACE pod-security.kubernetes.io/enforce=restricted`

The above command enforces the restricted policy for the NAMESPACE namespace.

You can also enable Pod Security Admission for all your namespaces. For example:

```
kubectl label --overwrite ns --all pod-security.kubernetes.io/warn=baseline
```

Default Value:

By default, there are no restrictions on the creation of privileged containers.

References:

1. https://kubernetes.io/docs/concepts/security/pod-security-admission/
2. https://aws.github.io/aws-eks-best-practices/pods/#restrict-the-containers-that-can-run-as-privileged

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

4.2.2 Minimize the admission of containers wishing to share the host process ID namespace (Manual)

Profile Applicability:

- Level 1

Description:

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

Rationale:

A container running in the host's PID namespace can inspect processes running outside the container. If the container also has access to ptrace capabilities this can be used to escalate privileges outside of the container.

There should be at least one admission control policy defined which does not permit containers to share the host PID namespace.

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

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostPID` containers

Search for the hostPID Flag: In the YAML output, look for the `hostPID` setting under the spec section to check if it is set to `true`.

```
kubectl get pods --all-namespaces -o json | jq -r '.items[] |
select(.spec.hostPID == true) | "\(.metadata.namespace)/\(.metadata.name)"'
```

OR

```
kubectl get pods --all-namespaces -o json | jq '.items[] |
select(.metadata.namespace != "kube-system" and .spec.hostPID == true) |
{pod: .metadata.name, namespace: .metadata.namespace, container:
.spec.containers[].name}'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command retrieves all pods across all namespaces in JSON format, then uses jq to filter out those with the `hostPID` flag set to `true`, and finally formats the output to show the namespace and name of each matching pod.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostPID` containers.

Default Value:

By default, there are no restrictions on the creation of `hostPID` containers.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

4.2.3 Minimize the admission of containers wishing to share the host IPC namespace (Manual)

Profile Applicability:

- Level 1

Description:

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

Rationale:

A container running in the host's IPC namespace can use IPC to interact with processes outside the container.

There should be at least one admission control policy defined which does not permit containers to share the host IPC namespace.

If you need to run containers which require `hostIPC`, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostIPC` containers

Search for the hostIPC Flag: In the YAML output, look for the `hostIPC` setting under the spec section to check if it is set to `true`.

```
kubectl get pods --all-namespaces -o json | jq -r '.items[] |
select(.spec.hostIPC == true) | "\(.metadata.namespace)/\(.metadata.name)"'
```

OR

```
kubectl get pods --all-namespaces -o json | jq '.items[] |
select(.metadata.namespace != "kube-system" and .spec.hostIPC == true) |
{pod: .metadata.name, namespace: .metadata.namespace, container:
.spec.containers[].name}'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command retrieves all pods across all namespaces in JSON format, then uses `jq` to filter out those with the `hostIPC` flag set to `true`, and finally formats the output to show the namespace and name of each matching pod.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostIPC` containers.

Default Value:

By default, there are no restrictions on the creation of `hostIPC` containers.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

4.2.4 Minimize the admission of containers wishing to share the host network namespace (Manual)

Profile Applicability:

- Level 1

Description:

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

Rationale:

A container running in the host's network namespace could access the local loopback device, and could access network traffic to and from other pods.

There should be at least one admission control policy defined which does not permit containers to share the host network namespace.

If you need to run containers which require access to the host's network namespaces, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostNetwork` containers

Given that manually checking each pod can be time-consuming, especially in large environments, you can use a more automated approach to filter out pods where `hostNetwork` is set to `true`. Here's a command using kubectl and jq:

```
kubectl get pods --all-namespaces -o json | jq -r '.items[] |
select(.spec.hostNetwork == true) |
"\(.metadata.namespace)/\(.metadata.name)"'
```

OR

```
kubectl get pods --all-namespaces -o json | jq '.items[] |
select(.metadata.namespace != "kube-system" and .spec.hostNetwork == true) |
{pod: .metadata.name, namespace: .metadata.namespace, container:
.spec.containers[].name}'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command retrieves all pods across all namespaces in JSON format, then uses jq to filter out those with the `hostNetwork` flag set to `true`, and finally formats the output to show the namespace and name of each matching pod.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostNetwork` containers.

Default Value:

By default, there are no restrictions on the creation of `hostNetwork` containers.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

4.2.5 Minimize the admission of containers with allowPrivilegeEscalation (Manual)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `allowPrivilegeEscalation` flag set to `true`. Allowing this right can lead to a process running a container getting more rights than it started with.

It's important to note that these rights are still constrained by the overall container sandbox, and this setting does not relate to the use of privileged containers.

Rationale:

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

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

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

Impact:

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

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of containers which allow privilege escalation.

This command gets all pods across all namespaces, outputs their details in JSON format, and uses jq to parse and filter the output for containers with `allowPrivilegeEscalation` set to `true`.

```
kubectl get pods --all-namespaces -o json | jq -r '.items[] |
select(any(.spec.containers[]; .securityContext.allowPrivilegeEscalation ==
true)) | "\(.metadata.namespace)/\(.metadata.name)"'
```

OR

```
kubectl get pods --all-namespaces -o json | jq '.items[] |
select(.metadata.namespace != "kube-system" and .spec.containers[];
.securityContext.allowPrivilegeEscalation == true) | {pod: .metadata.name,
namespace: .metadata.namespace, container: .spec.containers[].name}'
```

When creating a Pod Security Policy, ["kube-system"] namespaces are excluded by default.

This command uses `jq`, a command-line JSON processor, to parse the JSON output from kubectl get pods and filter out pods where any container has the securityContext.privileged flag set to true. Please note that you might need to adjust the command depending on your specific requirements and the structure of your pod specifications.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of containers with `.spec.allowPrivilegeEscalation` set to `true`.

Default Value:

By default, there are no restrictions on contained process ability to escalate privileges, within the context of the container.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

4.3 CNI Plugin

4.3.1 Ensure CNI plugin supports network policies. (Manual)

Profile Applicability:

- Level 1

Description:

There are a variety of CNI plugins available for Kubernetes. If the CNI in use does not support Network Policies it may not be possible to effectively restrict traffic in the cluster.

Rationale:

Kubernetes network policies are enforced by the CNI plugin in use. As such it is important to ensure that the CNI plugin supports both Ingress and Egress network policies.

Impact:

None.

Audit:

Review the documentation of CNI plugin in use by the cluster, and confirm that it supports network policies.

Remediation:

As with RBAC policies, network policies should adhere to the policy of least privileged access. Start by creating a deny all policy that restricts all inbound and outbound traffic from a namespace or create a global policy using Calico.

Default Value:

This will depend on the CNI plugin in use.

References:

1. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/
2. https://aws.github.io/aws-eks-best-practices/network/

Additional Information:

One example here is Flannel (https://github.com/flannel-io/flannel) which does not support Network policy unless Calico is also in use.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.4 Perform Automated Application Patch Management Perform application updates on enterprise assets through automated patch management on a monthly, or more frequent, basis.	●	●	●
v7	18.4 Only Use Up-to-date And Trusted Third-Party Components Only use up-to-date and trusted third-party components for the software developed by the organization.		●	●

4.3.2 Ensure that all Namespaces have Network Policies defined (Manual)

Profile Applicability:

- Level 1

Description:

Use network policies to isolate traffic in your cluster network.

Rationale:

Running different applications on the same Kubernetes cluster creates a risk of one compromised application attacking a neighboring application. Network segmentation is important to ensure that containers can communicate only with those they are supposed to. A network policy is a specification of how selections of pods are allowed to communicate with each other and other network endpoints.

Once there is any Network Policy in a namespace selecting a particular pod, that pod will reject any connections that are not allowed by any Network Policy. Other pods in the namespace that are not selected by any Network Policy will continue to accept all traffic"

Impact:

Once there is any Network Policy in a namespace selecting a particular pod, that pod will reject any connections that are not allowed by any Network Policy. Other pods in the namespace that are not selected by any Network Policy will continue to accept all traffic"

Audit:

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

```
kubectl get networkpolicy --all-namespaces
```

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

Remediation:

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

Default Value:

By default, network policies are not created.

References:

1. https://kubernetes.io/docs/concepts/services-networking/networkpolicies/
2. https://octetz.com/posts/k8s-network-policy-apis

3. https://kubernetes.io/docs/tasks/configure-pod-container/declare-network-policy/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	14.1 Segment the Network Based on Sensitivity Segment the network based on the label or classification level of the information stored on the servers, locate all sensitive information on separated Virtual Local Area Networks (VLANs).		●	●
v7	14.2 Enable Firewall Filtering Between VLANs Enable firewall filtering between VLANs to ensure that only authorized systems are able to communicate with other systems necessary to fulfill their specific responsibilities.		●	●

4.4 Secrets Management

4.4.1 Prefer using secrets as files over secrets as environment variables (Automated)

Profile Applicability:

- Level 1

Description:

Kubernetes supports mounting secrets as data volumes or as environment variables. Minimize the use of environment variable secrets.

Rationale:

It is reasonably common for application code to log out its environment (particularly in the event of an error). This will include any secret values passed in as environment variables, so secrets can easily be exposed to any user or entity who has access to the logs.

Impact:

Application code which expects to read secrets in the form of environment variables would need modification

Audit:

Run the following command to find references to objects which use environment variables defined from secrets.

```
kubectl get all -o jsonpath='{range .items[?(@..secretKeyRef)]} {.kind}
{.metadata.name} {"\n"}{end}' -A
```

Remediation:

If possible, rewrite application code to read secrets from mounted secret files, rather than from environment variables.

Default Value:

By default, secrets are not defined

References:

1. https://kubernetes.io/docs/concepts/configuration/secret/#using-secrets

Additional Information:

Mounting secrets as volumes has the additional benefit that secret values can be updated without restarting the pod

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).		●	●
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.		●	●
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

4.4.2 Consider external secret storage (Manual)

Profile Applicability:

- Level 1

Description:

Consider the use of an external secrets storage and management system, instead of using Kubernetes Secrets directly, if you have more complex secret management needs. Ensure the solution requires authentication to access secrets, has auditing of access to and use of secrets, and encrypts secrets. Some solutions also make it easier to rotate secrets.

Rationale:

Kubernetes supports secrets as first-class objects, but care needs to be taken to ensure that access to secrets is carefully limited. Using an external secrets provider can ease the management of access to secrets, especially where secrests are used across both Kubernetes and non-Kubernetes environments.

Impact:

None

Audit:

Review your secrets management implementation.

Remediation:

Refer to the secrets management options offered by your cloud provider or a third-party secrets management solution.

Default Value:

By default, no external secret management is configured.

CIS Controls:

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

Controls Version	Control	IG 1	IG 2	IG 3
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

4.5 General Policies

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

4.5.1 Create administrative boundaries between resources using namespaces (Manual)

Profile Applicability:

- Level 1

Description:

Use namespaces to isolate your Kubernetes objects.

Rationale:

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

Impact:

You need to switch between namespaces for administration.

Audit:

Run the below command and review the namespaces created in the cluster.

```
kubectl get namespaces
```

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

Remediation:

Follow the documentation and create namespaces for objects in your deployment as you need them.

Default Value:

By default, Kubernetes starts with four initial namespaces:

1. `default` - The default namespace for objects with no other namespace
2. `kube-system` - The namespace for objects created by the Kubernetes system
3. `kube-public` - The namespace for public-readable ConfigMap
4. `kube-node-lease` - The namespace for associated lease object for each node

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:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.8 Establish and Maintain Dedicated Computing Resources for All Administrative Work Establish and maintain dedicated computing resources, either physically or logically separated, for all administrative tasks or tasks requiring administrative access. The computing resources should be segmented from the enterprise's primary network and not be allowed internet access.			●
v7	12.1 Maintain an Inventory of Network Boundaries Maintain an up-to-date inventory of all of the organization's network boundaries.	●	●	●

4.5.2 The default namespace should not be used (Automated)

Profile Applicability:

- Level 1

Description:

Kubernetes provides a default namespace, where objects are placed if no namespace is specified for them. Placing objects in this namespace makes application of RBAC and other controls more difficult.

Rationale:

Resources in a Kubernetes cluster should be segregated by namespace, to allow for security controls to be applied at that level and to make it easier to manage resources.

Impact:

None

Audit:

Run this command to list objects in default namespace

```
kubectl get $(kubectl api-resources --verbs=list --namespaced=true -o name |
paste -sd, -) --ignore-not-found -n default
```

The only entries there should be system managed resources such as the `kubernetes` service

OR

```
kubectl get pods -n default
```

Returning `No resources found in default namespace.`

Remediation:

Ensure that namespaces are created to allow for appropriate segregation of Kubernetes resources and that all new resources are created in a specific namespace.

Default Value:

Unless a namespace is specific on object creation, the `default` namespace will be used

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.7 Use Standard Hardening Configuration Templates for Application Infrastructure Use standard, industry-recommended hardening configuration templates for application infrastructure components. This includes underlying servers, databases, and web servers, and applies to cloud containers, Platform as a Service (PaaS) components, and SaaS components. Do not allow in-house developed software to weaken configuration hardening.		●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

5 Managed services

This section consists of security recommendations for the Amazon EKS. These recommendations are applicable for configurations that Amazon EKS customers own and manage.

5.1 Image Registry and Image Scanning

This section contains recommendations relating to container image registries and securing images in those registries, such as Amazon Elastic Container Registry (ECR).

5.1.1 Ensure Image Vulnerability Scanning using Amazon ECR image scanning or a third party provider (Automated)

Profile Applicability:

- Level 1

Description:

Scan images being deployed to Amazon EKS for vulnerabilities.

Rationale:

Vulnerabilities in software packages can be exploited by hackers or malicious users to obtain unauthorized access to local cloud resources. Amazon ECR and other third party products allow images to be scanned for known vulnerabilities.

Impact:

If you are utilizing AWS ECR

The following are common image scan failures. You can view errors like this in the Amazon ECR console by displaying the image details or through the API or AWS CLI by using the DescribeImageScanFindings API.

UnsupportedImageError You may get an UnsupportedImageError error when attempting to scan an image that was built using an operating system that Amazon ECR doesn't support image scanning for. Amazon ECR supports package vulnerability scanning for major versions of Amazon Linux, Amazon Linux 2, Debian, Ubuntu, CentOS, Oracle Linux, Alpine, and RHEL Linux distributions. Amazon ECR does not support scanning images built from the Docker scratch image.

An UNDEFINED severity level is returned You may receive a scan finding that has a severity level of UNDEFINED. The following are the common causes for this:

The vulnerability was not assigned a priority by the CVE source.

The vulnerability was assigned a priority that Amazon ECR did not recognize.

To determine the severity and description of a vulnerability, you can view the CVE directly from the source.

Audit:

Please follow AWS ECS or your third party image scanning provider's guidelines for enabling Image Scanning.

```
aws ecr describe-repositories --repository-names $REPO_NAME --region
$REGION_CODE
```

Remediation:

To utilize AWS ECR for Image scanning please follow the steps below:

To create a repository configured for scan on push (AWS CLI)

```
aws ecr create-repository --repository-name $REPO_NAME --image-scanning-
configuration scanOnPush=true --region $REGION_CODE
```

To edit the settings of an existing repository (AWS CLI)

```
aws ecr put-image-scanning-configuration --repository-name $REPO_NAME --
image-scanning-configuration scanOnPush=true --region $REGION_CODE
```

Use the following steps to start a manual image scan using the AWS Management Console.

1. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/repositories.
2. From the navigation bar, choose the Region to create your repository in.
3. In the navigation pane, choose Repositories.
4. On the Repositories page, choose the repository that contains the image to scan.
5. On the Images page, select the image to scan and then choose Scan.

Default Value:

Images are not scanned by Default.

References:

1. https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.5 Perform Automated Vulnerability Scans of Internal Enterprise Assets Perform automated vulnerability scans of internal enterprise assets on a quarterly, or more frequent, basis. Conduct both authenticated and unauthenticated scans, using a SCAP-compliant vulnerability scanning tool.		●	●

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.6 Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets Perform automated vulnerability scans of externally-exposed enterprise assets using a SCAP-compliant vulnerability scanning tool. Perform scans on a monthly, or more frequent, basis.		●	●
v7	3.1 Run Automated Vulnerability Scanning Tools Utilize an up-to-date SCAP-compliant vulnerability scanning tool to automatically scan all systems on the network on a weekly or more frequent basis to identify all potential vulnerabilities on the organization's systems.		●	●
v7	3.2 Perform Authenticated Vulnerability Scanning Perform authenticated vulnerability scanning with agents running locally on each system or with remote scanners that are configured with elevated rights on the system being tested.		●	●

5.1.2 Minimize user access to Amazon ECR (Manual)

Profile Applicability:

- Level 1

Description:

Restrict user access to Amazon ECR, limiting interaction with build images to only authorized personnel and service accounts.

Rationale:

Weak access control to Amazon ECR may allow malicious users to replace built images with vulnerable containers.

Impact:

Care should be taken not to remove access to Amazon ECR for accounts that require this for their operation.

Audit:

Remediation:

Before you use IAM to manage access to Amazon ECR, you should understand what IAM features are available to use with Amazon ECR. To get a high-level view of how Amazon ECR and other AWS services work with IAM, see AWS Services That Work with IAM in the IAM User Guide.

Topics

- Amazon ECR Identity-Based Policies
- Amazon ECR Resource-Based Policies
- Authorization Based on Amazon ECR Tags
- Amazon ECR IAM Roles

Amazon ECR Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Amazon ECR supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the IAM User Guide.

Actions The Action element of an IAM identity-based policy describes the specific action or actions that will be allowed or denied by the policy. Policy actions usually have the same name as the associated AWS API operation. The action is used in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon ECR use the following prefix before the action: ecr:. For example, to grant someone permission to create an Amazon ECR repository with the Amazon ECR CreateRepository API operation, you include the ecr:CreateRepository action in their policy. Policy statements must include either an Action or NotAction element. Amazon ECR defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as follows:

```
"Action": [ "ecr:action1", "ecr:action2"
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word Describe, include the following action:

```
"Action": "ecr:Describe*"
```

To see a list of Amazon ECR actions, see Actions, Resources, and Condition Keys for Amazon Elastic Container Registry in the IAM User Guide.

Resources The Resource element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. You specify a resource using an ARN or using the wildcard (*) to indicate that the statement applies to all resources.

An Amazon ECR repository resource has the following ARN:

```
arn:${Partition}:ecr:${Region}:${Account}:repository/${Repository-
name}
```

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, to specify the my-repo repository in the us-east-1 Region in your statement, use the following ARN:

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/my-repo"
```

To specify all repositories that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:ecr:us-east-1:123456789012:repository/*"
```

To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [ "resource1", "resource2"
```

To see a list of Amazon ECR resource types and their ARNs, see Resources Defined by Amazon Elastic Container Registry in the IAM User Guide. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon Elastic Container Registry.

Condition Keys The Condition element (or Condition block) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can build conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM Policy Elements: Variables and Tags in the IAM User Guide.

Amazon ECR defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see AWS Global Condition Context Keys in the IAM User Guide.

Most Amazon ECR actions support the aws:ResourceTag and ecr:ResourceTag condition keys. For more information, see Using Tag-Based Access Control.

To see a list of Amazon ECR condition keys, see Condition Keys Defined by Amazon Elastic Container Registry in the IAM User Guide. To learn with which actions and resources you can use a condition key, see Actions Defined by Amazon Elastic Container Registry.

References:

1. https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning.html#scanning-repository

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

5.1.3 Minimize cluster access to read-only for Amazon ECR (Manual)

Profile Applicability:

- Level 1

Description:

Configure the Cluster Service Account with Storage Object Viewer Role to only allow read-only access to Amazon ECR.

Rationale:

The Cluster Service Account does not require administrative access to Amazon ECR, only requiring pull access to containers to deploy onto Amazon EKS. Restricting permissions follows the principles of least privilege and prevents credentials from being abused beyond the required role.

Impact:

A separate dedicated service account may be required for use by build servers and other robot users pushing or managing container images.

Audit:

Review AWS ECS worker node IAM role (NodeInstanceRole) IAM Policy Permissions to verify that they are set and the minimum required level.

If utilizing a 3rd party tool to scan images utilize the minimum required permission level required to interact with the cluster - generally this should be read-only.

Remediation:

You can use your Amazon ECR images with Amazon EKS, but you need to satisfy the following prerequisites.

The Amazon EKS worker node IAM role (NodeInstanceRole) that you use with your worker nodes must possess the following IAM policy permissions for Amazon ECR.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

Default Value:

If you used eksctl or the AWS CloudFormation templates in Getting Started with Amazon EKS to create your cluster and worker node groups, these IAM permissions are applied to your worker node IAM role by default.

References:

1. https://docs.aws.amazon.com/AmazonECR/latest/userguide/ECR_on_EKS.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

5.1.4 Minimize Container Registries to only those approved (Manual)

Profile Applicability:

- Level 1

Description:

Use approved container registries.

Rationale:

Allowing unrestricted access to external container registries provides the opportunity for malicious or unapproved containers to be deployed into the cluster. Allowlisting only approved container registries reduces this risk.

Impact:

All container images to be deployed to the cluster must be hosted within an approved container image registry.

Audit:

Remediation:

To minimize AWS ECR container registries to only those approved, you can follow these steps:

1. Define your approval criteria: Determine the criteria that containers must meet to be considered approved. This can include factors such as security, compliance, compatibility, and other requirements.
2. Identify all existing ECR registries: Identify all ECR registries that are currently being used in your organization.
3. Evaluate ECR registries against approval criteria: Evaluate each ECR registry against your approval criteria to determine whether it should be approved or not. This can be done by reviewing the registry settings and configuration, as well as conducting security assessments and vulnerability scans.
4. Establish policies and procedures: Establish policies and procedures that outline how ECR registries will be approved, maintained, and monitored. This should include guidelines for developers to follow when selecting a registry for their container images.
5. Implement access controls: Implement access controls to ensure that only approved ECR registries are used to store and distribute container images. This can be done by setting up IAM policies and roles that restrict access to unapproved registries or create a whitelist of approved registries.
6. Monitor and review: Continuously monitor and review the use of ECR registries to ensure that they continue to meet your approval criteria. This can include

regularly reviewing access logs, scanning for vulnerabilities, and conducting periodic audits.

By following these steps, you can minimize AWS ECR container registries to only those approved, which can help to improve security, reduce complexity, and streamline container management in your organization. Additionally, AWS provides several tools and services that can help you manage your ECR registries, such as AWS Config, AWS CloudFormation, and AWS Identity and Access Management (IAM).

Default Value:

Container registries are not restricted by default and Kubernetes assumes your default CR is Docker Hub.

References:

1. https://aws.amazon.com/blogs/opensource/using-open-policy-agent-on-amazon-eks/

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	2.5 Allowlist Authorized Software Use technical controls, such as application allowlisting, to ensure that only authorized software can execute or be accessed. Reassess bi-annually, or more frequently.		●	●
v7	5.3 Securely Store Master Images Store the master images and templates on securely configured servers, validated with integrity monitoring tools, to ensure that only authorized changes to the images are possible.		●	●
v7	13.4 Only Allow Access to Authorized Cloud Storage or Email Providers Only allow access to authorized cloud storage or email providers.		●	●

5.2 Identity and Access Management (IAM)

This section contains recommendations relating to using AWS IAM with Amazon EKS.

5.2.1 Prefer using dedicated EKS Service Accounts (Automated)

Profile Applicability:

- Level 1

Description:

Kubernetes workloads should not use cluster node service accounts to authenticate to Amazon EKS APIs. Each Kubernetes workload that needs to authenticate to other AWS services using AWS IAM should be provisioned with a dedicated Service account.

Rationale:

Manual approaches for authenticating Kubernetes workloads running on Amazon EKS against AWS APIs are: storing service account keys as a Kubernetes secret (which introduces manual key rotation and potential for key compromise); or use of the underlying nodes' IAM Service account, which violates the principle of least privilege on a multi-tenanted node, when one pod needs to have access to a service, but every other pod on the node that uses the Service account does not.

Audit:

For each namespace in the cluster, review the rights assigned to the default service account and ensure that it has no roles or cluster roles bound to it apart from the defaults.

Additionally ensure that the automountServiceAccountToken: false setting is in place for each default service account.

Remediation:

With IAM roles for service accounts on Amazon EKS clusters, you can associate an IAM role with a Kubernetes service account. This service account can then provide AWS permissions to the containers in any pod that uses that service account. With this feature, you no longer need to provide extended permissions to the worker node IAM role so that pods on that node can call AWS APIs.

Applications must sign their AWS API requests with AWS credentials. This feature provides a strategy for managing credentials for your applications, similar to the way that Amazon EC2 instance profiles provide credentials to Amazon EC2 instances. Instead of creating and distributing your AWS credentials to the containers or using the Amazon EC2 instance’s role, you can associate an IAM role with a Kubernetes service account. The applications in the pod’s containers can then use an AWS SDK or the AWS CLI to make API requests to authorized AWS services.

The IAM roles for service accounts feature provides the following benefits:

- Least privilege — By using the IAM roles for service accounts feature, you no longer need to provide extended permissions to the worker node IAM role so that pods on that node can call AWS APIs. You can scope IAM permissions to a service account, and only pods that use that service account have access to those permissions. This feature also eliminates the need for third-party solutions such as kiam or kube2iam.

- Credential isolation — A container can only retrieve credentials for the IAM role that is associated with the service account to which it belongs. A container never has access to credentials that are intended for another container that belongs to another pod.
- Audit-ability — Access and event logging is available through CloudTrail to help ensure retrospective auditing.

To get started, see list text hereEnabling IAM roles for service accounts on your cluster.

For an end-to-end walkthrough using eksctl, see Walkthrough: Updating a DaemonSet to use IAM for service accounts.

References:

1. https://docs.aws.amazon.com/eks/latest/userguide/iam-roles-for-service-accounts.html
2. https://docs.aws.amazon.com/eks/latest/userguide/iam-roles-for-service-accounts-cni-walkthrough.html
3. https://aws.github.io/aws-eks-best-practices/security/docs/iam/#scope-the-iam-role-trust-policy-for-irsa-to-the-service-account-name

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.8 Establish and Maintain Dedicated Computing Resources for All Administrative Work Establish and maintain dedicated computing resources, either physically or logically separated, for all administrative tasks or tasks requiring administrative access. The computing resources should be segmented from the enterprise's primary network and not be allowed internet access.			●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.	●	●	●

5.3 AWS EKS Key Management Service

This section contains recommendations relating to using AWS EKS Key Management.

5.3.1 Ensure Kubernetes Secrets are encrypted using Customer Master Keys (CMKs) managed in AWS KMS (Manual)

Profile Applicability:

- Level 1

Description:

Encrypt Kubernetes secrets, stored in etcd, using secrets encryption feature during Amazon EKS cluster creation.

Rationale:

Kubernetes can store secrets that pods can access via a mounted volume. Today, Kubernetes secrets are stored with Base64 encoding, but encrypting is the recommended approach. Amazon EKS clusters version 1.13 and higher support the capability of encrypting your Kubernetes secrets using AWS Key Management Service (KMS) Customer Managed Keys (CMK). The only requirement is to enable the encryption provider support during EKS cluster creation.

Use AWS Key Management Service (KMS) keys to provide envelope encryption of Kubernetes secrets stored in Amazon EKS. Implementing envelope encryption is considered a security best practice for applications that store sensitive data and is part of a defense in depth security strategy.

Application-layer Secrets Encryption provides an additional layer of security for sensitive data, such as user defined Secrets and Secrets required for the operation of the cluster, such as service account keys, which are all stored in etcd.

Using this functionality, you can use a key, that you manage in AWS KMS, to encrypt data at the application layer. This protects against attackers in the event that they manage to gain access to etcd.

Audit:

For Amazon EKS clusters with Secrets Encryption enabled, run the AWS CLI command:

```
aws eks describe-cluster --name=<cluster-name>
```

From the output of the command, search if the following configuration exist with valid AWS `keyArn` :

```
"encryptionConfig": [
         {
            "provider": {
               "keyArn": "string"
            },
            "resources": [ "string" ]
         }
      ],
```

Remediation:

This process can only be performed during Cluster Creation.

Enable 'Secrets Encryption' during Amazon EKS cluster creation as described in the links within the 'References' section.

Default Value:

By default secrets created using the Kubernetes API are stored in *tmpfs* and are encrypted at rest.

References:

1. https://aws.amazon.com/about-aws/whats-new/2020/03/amazon-eks-adds-envelope-encryption-for-secrets-with-aws-kms/
2. https://docs.aws.amazon.com/eks/latest/APIReference/API_DescribeCluster.html

CIS Controls:

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

5.4 Cluster Networking

This section contains recommendations relating to network security configurations in Amazon EKS.

5.4.1 Restrict Access to the Control Plane Endpoint (Automated)

Profile Applicability:

- Level 1

Description:

Enable Endpoint Private Access to restrict access to the cluster's control plane to only an allowlist of authorized IPs.

Rationale:

Authorized networks are a way of specifying a restricted range of IP addresses that are permitted to access your cluster's control plane. Kubernetes Engine uses both Transport Layer Security (TLS) and authentication to provide secure access to your cluster's control plane from the public internet. This provides you the flexibility to administer your cluster from anywhere; however, you might want to further restrict access to a set of IP addresses that you control. You can set this restriction by specifying an authorized network.

Restricting access to an authorized network can provide additional security benefits for your container cluster, including:

- Better protection from outsider attacks: Authorized networks provide an additional layer of security by limiting external access to a specific set of addresses you designate, such as those that originate from your premises. This helps protect access to your cluster in the case of a vulnerability in the cluster's authentication or authorization mechanism.
- Better protection from insider attacks: Authorized networks help protect your cluster from accidental leaks of master certificates from your company's premises. Leaked certificates used from outside Cloud Services and outside the authorized IP ranges (for example, from addresses outside your company) are still denied access.

Impact:

When implementing Endpoint Private Access, be careful to ensure all desired networks are on the allowlist (whitelist) to prevent inadvertently blocking external access to your cluster's control plane.

Audit:

Check for the following to be 'enabled: true'

```
export CLUSTER_NAME=<your cluster name>
export REGION_CODE=<your region>

aws eks describe-cluster \
  --name "${CLUSTER_NAME}" \
  --region "${REGION_CODE}" \
  --query
"cluster.resourcesVpcConfig.{endpointPrivateAccess:endpointPrivateAccess,endp
ointPublicAccess:endpointPublicAccess,publicAccessCidrs:publicAccessCidrs}" \
  --output json

Check output for:
    "endpointPrivateAccess": true,
    "endpointPublicAccess": true
```

Check for publicAccessCidrs is to a valid IP address not set to 0.0.0.0/0:

```
    "publicAccessCidrs": [
        "203.0.113.5/32"
    ]
```

Remediation:

By enabling private endpoint access to the Kubernetes API server, all communication between your nodes and the API server stays within your VPC. You can also limit the IP addresses that can access your API server from the internet, or completely disable internet access to the API server.

With this in mind, you can update your cluster accordingly using the AWS CLI to ensure that Private Endpoint Access is enabled.

If you choose to also enable Public Endpoint Access then you should also configure a list of allowable CIDR blocks, resulting in restricted access from the internet. If you specify no CIDR blocks, then the public API server endpoint is able to receive and process requests from all IP addresses by defaulting to ['0.0.0.0/0'].

For example, the following command would enable private access to the Kubernetes API as well as limited public access over the internet from a single IP address (noting the /32 CIDR suffix):

```
aws eks update-cluster-config --region $REGION_CODE --name
$CLUSTER_NAME --resources-vpc-config endpointPrivateAccess=true,
endpointPublicAccess=true, publicAccessCidrs="203.0.113.5/32"
```

Note:

The CIDR blocks specified cannot include reserved addresses. There is a maximum number of CIDR blocks that you can specify. For more information, see the EKS Service Quotas link in the references section. For more detailed information, see the EKS Cluster Endpoint documentation link in the references section.

Default Value:

By default, Endpoint Public Access is disabled.

References:

1. https://docs.aws.amazon.com/eks/latest/userguide/cluster-endpoint.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v8	9.3 Maintain and Enforce Network-Based URL Filters Enforce and update network-based URL filters to limit an enterprise asset from connecting to potentially malicious or unapproved websites. Example implementations include category-based filtering, reputation-based filtering, or through the use of block lists. Enforce filters for all enterprise assets.		●	●
v7	7.4 Maintain and Enforce Network-Based URL Filters Enforce network-based URL filters that limit a system's ability to connect to websites not approved by the organization. This filtering shall be enforced for each of the organization's systems, whether they are physically at an organization's facilities or not.		●	●

5.4.2 Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Automated)

Profile Applicability:

- Level 1

Description:

Disable access to the Kubernetes API from outside the node network if it is not required.

Rationale:

In a private cluster, the master node has two endpoints, a private and public endpoint. The private endpoint is the internal IP address of the master, behind an internal load balancer in the master's VPC network. Nodes communicate with the master using the private endpoint. The public endpoint enables the Kubernetes API to be accessed from outside the master's VPC network.

Although Kubernetes API requires an authorized token to perform sensitive actions, a vulnerability could potentially expose the Kubernetes publicly with unrestricted access. Additionally, an attacker may be able to identify the current cluster and Kubernetes API version and determine whether it is vulnerable to an attack. Unless required, disabling public endpoint will help prevent such threats, and require the attacker to be on the master's VPC network to perform any attack on the Kubernetes API.

Impact:

Configure the EKS cluster endpoint to be private.

1. Leave the cluster endpoint public and specify which CIDR blocks can communicate with the cluster endpoint. The blocks are effectively a whitelisted set of public IP addresses that are allowed to access the cluster endpoint.
2. Configure public access with a set of whitelisted CIDR blocks and set private endpoint access to enabled. This will allow public access from a specific range of public IPs while forcing all network traffic between the kubelets (workers) and the Kubernetes API through the cross-account ENIs that get provisioned into the cluster VPC when the control plane is provisioned.

Audit:

Check for private endpoint access to the Kubernetes API server

```
export CLUSTER_NAME=<your cluster name>
export REGION_CODE=<your region>

aws eks describe-cluster \
  --name "${CLUSTER_NAME}" \
  --region "${REGION_CODE}" \
  --query
"cluster.resourcesVpcConfig.{endpointPublicAccess:endpointPublicAccess,
endpointPrivateAccess:endpointPrivateAccess}" \
  --output json

Check for the following to be '"endpointPrivateAccess": true'
Check for the following to be '"endpointPublicAccess": true'
```

Remediation:

By enabling private endpoint access to the Kubernetes API server, all communication between your nodes and the API server stays within your VPC.

With this in mind, you can update your cluster accordingly using the AWS CLI to ensure that Private Endpoint Access is enabled.

For example, the following command would enable private access to the Kubernetes API and ensure that no public access is permitted:

```
aws eks update-cluster-config --region $AWS_REGION --name
$CLUSTER_NAME --resources-vpc-config
endpointPrivateAccess=true,endpointPublicAccess=false
```

Note: For more detailed information, see the EKS Cluster Endpoint documentation link in the references section.

Default Value:

By default, the Public Endpoint is disabled.

References:

1. https://docs.aws.amazon.com/eks/latest/userguide/cluster-endpoint.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	12 Boundary Defense Boundary Defense			

5.4.3 Ensure clusters are created with Private Nodes (Automated)

Profile Applicability:

- Level 1

Description:

Disable public IP addresses for cluster nodes, so that they only have private IP addresses. Private Nodes are nodes with no public IP addresses.

Rationale:

Disabling public IP addresses on cluster nodes restricts access to only internal networks, forcing attackers to obtain local network access before attempting to compromise the underlying Kubernetes hosts.

Impact:

To enable Private Nodes, the cluster has to also be configured with a private master IP range and IP Aliasing enabled.

Private Nodes do not have outbound access to the public internet. If you want to provide outbound Internet access for your private nodes, you can use Cloud NAT or you can manage your own NAT gateway.

Audit:

Check for the following are 'enabled: true'

```
export CLUSTER_NAME=<your cluster name>
aws eks describe-cluster --name ${CLUSTER_NAME} --query
"cluster.resourcesVpcConfig.endpointPrivateAccess"
aws eks describe-cluster --name ${CLUSTER_NAME} --query
"cluster.resourcesVpcConfig.endpointPublicAccess"
```

Check for the following is not null and set with appropriate IP and not 0.0.0.0/0:

```
export CLUSTER_NAME=<your cluster name>
aws eks describe-cluster --name ${CLUSTER_NAME} --query
"cluster.resourcesVpcConfig.publicAccessCidrs"
```

Note: In addition include the check if the nodes are deployed in private subnets and no public IP is assigned. The private subnets should not be associated with a route table that has a route to an Internet Gateway (IGW).

Remediation:

To disable public IP addresses for EKS nodegroup nodes using the AWS CLI, you must ensure the following when running create-nodegroup:

- Use private subnets (that don't auto-assign public IPs).
- Set associatePublicIpAddress to false.

```
"NetworkInterfaces": [{
  "AssociatePublicIpAddress": false
}]
```

You can restrict access to the control plane endpoint using:

```
aws eks update-cluster-config \
  --name $CLUSTER_NAME \
  --region $REGION_CODE \
  --resources-vpc-config endpointPublicAccess=false,
endpointPrivateAccess=true
```

This makes the API server private, but does not affect node IPs.

To ensure nodes use only private IPs:

- Use aws eks create-nodegroup with only private subnets, or
- Use a launch template with AssociatePublicIpAddress=false.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	12 Boundary Defense Boundary Defense			

5.4.4 Ensure AmazonEKSNetworkingPolicy is Enabled and set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

AmazonEKSNetworkingPolicy is an AWS-managed add-on that provides native support for Kubernetes NetworkPolicy enforcement within Amazon Elastic Kubernetes Service (EKS) clusters. It enables organizations to define and apply fine-grained, pod-level network access controls that regulate traffic between workloads and external endpoints. Built on an AWS-optimized implementation of Calico, the add-on integrates seamlessly with the EKS control plane to deliver a secure and scalable approach to network segmentation without requiring manual CNI plugin installation. By leveraging AmazonEKSNetworkingPolicy, enterprises can strengthen their cluster security posture, enforce zero-trust networking principles, and ensure compliance with organizational and regulatory access control standards.

Rationale:

By default, all pod to pod traffic within a cluster is allowed. Network Policy creates a pod-level firewall that can be used to restrict traffic between sources. Pod traffic is restricted by having a Network Policy that selects it (through the use of labels). Once there is any Network Policy in a namespace selecting a particular pod, that pod will reject any connections that are not allowed by any Network Policy. Other pods in the namespace that are not selected by any Network Policy will continue to accept all traffic.

Implementing **AmazonEKSNetworkingPolicy** provides significant security and operational benefits for Amazon EKS environments by introducing native, AWS-managed enforcement of Kubernetes NetworkPolicies. This capability allows organizations to implement **fine-grained, pod-level network segmentation**, reducing the risk of lateral movement and unauthorized communication within clusters. By leveraging an **AWS-optimized Calico engine**, AmazonEKSNetworkingPolicy eliminates the complexity of manually deploying and managing third-party CNIs while ensuring consistent policy enforcement aligned with Kubernetes standards. The add-on enhances compliance with **zero-trust networking principles**, supports regulatory frameworks that require network isolation, and provides centralized visibility and control over intra-cluster communication. Ultimately, it helps organizations improve their **security posture**, **reduce operational overhead**, and **simplify governance** in modern containerized environments.

Impact:

The implementation of **AmazonEKSNetworkingPolicy** has a substantial impact on the overall **security, compliance, and operational efficiency** of Amazon EKS environments. By enabling native enforcement of Kubernetes NetworkPolicies, it strengthens the cluster's **defense-in-depth** strategy through precise control of pod-to-pod and pod-to-external communication. This reduces the attack surface, mitigates the risk of lateral movement, and ensures that only explicitly authorized traffic is permitted within the cluster. From a compliance perspective, it supports adherence to **zero-trust security frameworks** and organizational policies that mandate network segmentation and isolation. Operationally, AmazonEKSNetworkingPolicy simplifies the management of network security by providing an AWS-managed, scalable, and fully integrated solution—allowing teams to focus on application innovation rather than complex network configurations.

Note that enabling Network Policy enforcement consumes additional resources in nodes. Specifically, it increases the memory footprint of the kube-system process by approximately 128MB, and requires approximately 300 millicores of CPU.

Audit:

Check for the following is true:

```
export CLUSTER_NAME=<your cluster name>
aws eks describe-addon --cluster-name ${CLUSTER_NAME} --addon-name vpc-cni --
query addon.configurationValues

Output should read:
"{\"enableNetworkPolicy\":\"true\"}"
```

Remediation:

Make sure Amazon VPC CNI is added and vpc-cni is active and upgraded to appropriate version in clusters Add-Ons. To update add-on run the following statement at the aws cli:

```
aws eks update-addon --cluster-name $CLUSTER_NAME --addon-name vpc-cni --
configuration-values '{"enableNetworkPolicy":"true"}'
```

Default Value:

By default, Network Policy is disabled.

References:

1. https://docs.aws.amazon.com/eks/latest/userguide/eks-networking-add-ons.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.6 Use of Secure Network Management and Communication Protocols Use secure network management and communication protocols (e.g., 802.1X, Wi-Fi Protected Access 2 (WPA2) Enterprise or greater).		●	●
v7	9.2 Ensure Only Approved Ports, Protocols and Services Are Running Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.		●	●
v7	9.4 Apply Host-based Firewalls or Port Filtering Apply host-based firewalls or port filtering tools on end systems, with a default-deny rule that drops all traffic except those services and ports that are explicitly allowed.	●	●	●

5.4.5 Encrypt traffic to HTTPS load balancers with TLS certificates (Manual)

Profile Applicability:

- Level 1

Description:

Encrypt traffic to HTTPS load balancers using TLS certificates.

Rationale:

Encrypting traffic between users and your Kubernetes workload is fundamental to protecting data sent over the web.

Audit:

Your load balancer vendor can provide details on auditing the certificates and policies required to utilize TLS.

Remediation:

Your load balancer vendor can provide details on configuring HTTPS with TLS.

References:

1. https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/data-protection.html
2. https://docs.aws.amazon.com/elasticloadbalancing/latest/application/create-https-listener.html

CIS Controls:

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

5.5 Authentication and Authorization

This section contains recommendations relating to authentication and authorization in Amazon EKS.

5.5.1 Manage Kubernetes RBAC users with AWS IAM Authenticator for Kubernetes or Upgrade to AWS CLI v1.16.156 or greater (Manual)

Profile Applicability:

- Level 1

Description:

Amazon EKS uses IAM to provide authentication to your Kubernetes cluster through the AWS IAM Authenticator for Kubernetes. You can configure the stock kubectl client to work with Amazon EKS by installing the AWS IAM Authenticator for Kubernetes and modifying your kubectl configuration file to use it for authentication.

Rationale:

On- and off-boarding users is often difficult to automate and prone to error. Using a single source of truth for user permissions reduces the number of locations that an individual must be off-boarded from, and prevents users gaining unique permissions sets that increase the cost of audit.

Impact:

Users must now be assigned to the IAM group created to use this namespace and deploy applications. If they are not they will not be able to access the namespace or deploy.

Audit:

To Audit access to the namespace $NAMESPACE, assume the IAM role yourIAMRoleName for a user that you created, and then run the following command:

```
$ kubectl get role -n $NAMESPACE
```

The response lists the RBAC role that has access to this namespace.

Remediation:

Refer to the 'Managing users or IAM roles for your cluster' in Amazon EKS documentation.

Note: If using AWS CLI version 1.16.156 or later there is no need to install the AWS IAM Authenticator anymore.

The relevant AWS CLI commands, depending on the use case, are:

```
aws eks update-kubeconfig
aws eks get-token
```

Default Value:

For role-based access control (RBAC), system:masters permissions are configured in the Amazon EKS control plane

References:

1. https://docs.aws.amazon.com/eks/latest/userguide/add-user-role.html

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	16.2 Configure Centralized Point of Authentication Configure access for all accounts through as few centralized points of authentication as possible, including network, security, and cloud systems.		●	●

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**Control Plane Components**		
2	**Control Plane Configuration**		
2.1	**Logging**		
2.1.1	Enable audit Logs (Automated)	☐	☐
2.1.2	Ensure audit logs are collected and managed (Manual)	☐	☐
3	**Worker Nodes**		
3.1	**Worker Node Configuration Files**		
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive (Automated)	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root (Automated)	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Automated)	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root (Automated)	☐	☐
3.2	**Kubelet**		
3.2.1	Ensure that the Anonymous Auth is Not Enabled (Automated)	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	☐	☐
3.2.3	Ensure that a Client CA File is Configured (Automated)	☐	☐
3.2.4	Ensure that the --read-only-port is disabled (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated)	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true (Automated)	☐	☐
3.2.7	Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture (Automated)	☐	☐
3.2.8	Ensure that the --rotate-certificates argument is not present or is set to true (Automated)	☐	☐
3.2.9	Ensure that the RotateKubeletServerCertificate argument is set to true (Automated)	☐	☐
4	**Policies**		
4.1	**RBAC and Service Accounts**		
4.1.1	Ensure that the cluster-admin role is only used where required (Manual)	☐	☐
4.1.2	Minimize access to secrets (Manual)	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles (Manual)	☐	☐
4.1.4	Minimize access to create pods (Manual)	☐	☐
4.1.5	Ensure that default service accounts are not actively used. (Manual)	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary (Manual)	☐	☐
4.1.7	Cluster Access Manager API to streamline and enhance the management of access controls within EKS clusters (Automated)	☐	☐
4.1.8	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.1.9	Minimize access to create persistent volumes (Manual)	☐	☐
4.1.10	Minimize access to the proxy sub-resource of nodes (Manual)	☐	☐
4.1.11	Minimize access to webhook configuration objects (Manual)	☐	☐
4.1.12	Minimize access to the service account token creation (Manual)	☐	☐
4.2	**Pod Security Standards**		
4.2.1	Minimize the admission of privileged containers (Manual)	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace (Manual)	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace (Manual)	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace (Manual)	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation (Manual)	☐	☐
4.3	**CNI Plugin**		
4.3.1	Ensure CNI plugin supports network policies. (Manual)	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined (Manual)	☐	☐
4.4	**Secrets Management**		
4.4.1	Prefer using secrets as files over secrets as environment variables (Automated)	☐	☐
4.4.2	Consider external secret storage (Manual)	☐	☐
4.5	**General Policies**		

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.5.1	Create administrative boundaries between resources using namespaces (Manual)	☐	☐
4.5.2	The default namespace should not be used (Automated)	☐	☐
5	**Managed services**		
5.1	**Image Registry and Image Scanning**		
5.1.1	Ensure Image Vulnerability Scanning using Amazon ECR image scanning or a third party provider (Automated)	☐	☐
5.1.2	Minimize user access to Amazon ECR (Manual)	☐	☐
5.1.3	Minimize cluster access to read-only for Amazon ECR (Manual)	☐	☐
5.1.4	Minimize Container Registries to only those approved (Manual)	☐	☐
5.2	**Identity and Access Management (IAM)**		
5.2.1	Prefer using dedicated EKS Service Accounts (Automated)	☐	☐
5.3	**AWS EKS Key Management Service**		
5.3.1	Ensure Kubernetes Secrets are encrypted using Customer Master Keys (CMKs) managed in AWS KMS (Manual)	☐	☐
5.4	**Cluster Networking**		
5.4.1	Restrict Access to the Control Plane Endpoint (Automated)	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Automated)	☐	☐
5.4.3	Ensure clusters are created with Private Nodes (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.4.4	Ensure AmazonEKSNetworkingPolicy is Enabled and set as appropriate (Automated)	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates (Manual)	☐	☐
5.5	**Authentication and Authorization**		
5.5.1	Manage Kubernetes RBAC users with AWS IAM Authenticator for Kubernetes or Upgrade to AWS CLI v1.16.156 or greater (Manual)	☐	☐

Appendix: CIS Controls v7 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Enable audit Logs	☐	☐
2.1.2	Ensure audit logs are collected and managed	☐	☐
3.2.1	Ensure that the Anonymous Auth is Not Enabled	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.7	Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.5.1	Create administrative boundaries between resources using namespaces	☐	☐
4.5.2	The default namespace should not be used	☐	☐
5.1.2	Minimize user access to Amazon ECR	☐	☐
5.1.3	Minimize cluster access to read-only for Amazon ECR	☐	☐
5.2.1	Prefer using dedicated EKS Service Accounts	☐	☐
5.4.4	Ensure AmazonEKSNetworkingPolicy is Enabled and set as appropriate	☐	☐

Appendix: CIS Controls v7 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Enable audit Logs	☐	☐
2.1.2	Ensure audit logs are collected and managed	☐	☐
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
3.2.1	Ensure that the Anonymous Auth is Not Enabled	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.3	Ensure that a Client CA File is Configured	☐	☐
3.2.4	Ensure that the --read-only-port is disabled	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture	☐	☐
3.2.8	Ensure that the --rotate-certificates argument is not present or is set to true	☐	☐
3.2.9	Ensure that the RotateKubeletServerCertificate argument is set to true	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Minimize access to create pods	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure CNI plugin supports network policies.	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.5.1	Create administrative boundaries between resources using namespaces	☐	☐
4.5.2	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning using Amazon ECR image scanning or a third party provider	☐	☐
5.1.2	Minimize user access to Amazon ECR	☐	☐
5.1.3	Minimize cluster access to read-only for Amazon ECR	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.2.1	Prefer using dedicated EKS Service Accounts	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.4	Ensure AmazonEKSNetworkingPolicy is Enabled and set as appropriate	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates	☐	☐
5.5.1	Manage Kubernetes RBAC users with AWS IAM Authenticator for Kubernetes or Upgrade to AWS CLI v1.16.156 or greater	☐	☐

Appendix: CIS Controls v7 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Enable audit Logs	☐	☐
2.1.2	Ensure audit logs are collected and managed	☐	☐
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
3.2.1	Ensure that the Anonymous Auth is Not Enabled	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.3	Ensure that a Client CA File is Configured	☐	☐
3.2.4	Ensure that the --read-only-port is disabled	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture	☐	☐
3.2.8	Ensure that the --rotate-certificates argument is not present or is set to true	☐	☐
3.2.9	Ensure that the RotateKubeletServerCertificate argument is set to true	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Minimize access to create pods	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.1.7	Cluster Access Manager API to streamline and enhance the management of access controls within EKS clusters	☐	☐
4.1.8	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure CNI plugin supports network policies.	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.4.2	Consider external secret storage	☐	☐
4.5.1	Create administrative boundaries between resources using namespaces	☐	☐
4.5.2	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning using Amazon ECR image scanning or a third party provider	☐	☐
5.1.2	Minimize user access to Amazon ECR	☐	☐
5.1.3	Minimize cluster access to read-only for Amazon ECR	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.2.1	Prefer using dedicated EKS Service Accounts	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted using Customer Master Keys (CMKs) managed in AWS KMS	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.4.4	Ensure AmazonEKSNetworkingPolicy is Enabled and set as appropriate	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates	☐	☐
5.5.1	Manage Kubernetes RBAC users with AWS IAM Authenticator for Kubernetes or Upgrade to AWS CLI v1.16.156 or greater	☐	☐

Appendix: CIS Controls v7 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
4.1.9	Minimize access to create persistent volumes	☐	☐
4.1.10	Minimize access to the proxy sub-resource of nodes	☐	☐
4.1.11	Minimize access to webhook configuration objects	☐	☐
4.1.12	Minimize access to the service account token creation	☐	☐

Appendix: CIS Controls v8 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Enable audit Logs	☐	☐
2.1.2	Ensure audit logs are collected and managed	☐	☐
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
3.2.1	Ensure that the Anonymous Auth is Not Enabled	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.7	Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.3.1	Ensure CNI plugin supports network policies.	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
5.1.2	Minimize user access to Amazon ECR	☐	☐
5.1.3	Minimize cluster access to read-only for Amazon ECR	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐

Appendix: CIS Controls v8 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Enable audit Logs	☐	☐
2.1.2	Ensure audit logs are collected and managed	☐	☐
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
3.2.1	Ensure that the Anonymous Auth is Not Enabled	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.3	Ensure that a Client CA File is Configured	☐	☐
3.2.4	Ensure that the --read-only-port is disabled	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture	☐	☐
3.2.8	Ensure that the --rotate-certificates argument is not present or is set to true	☐	☐
3.2.9	Ensure that the RotateKubeletServerCertificate argument is set to true	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.1.7	Cluster Access Manager API to streamline and enhance the management of access controls within EKS clusters	☐	☐
4.1.8	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure CNI plugin supports network policies.	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.4.2	Consider external secret storage	☐	☐
4.5.2	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning using Amazon ECR image scanning or a third party provider	☐	☐
5.1.2	Minimize user access to Amazon ECR	☐	☐
5.1.3	Minimize cluster access to read-only for Amazon ECR	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted using Customer Master Keys (CMKs) managed in AWS KMS	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.4.4	Ensure AmazonEKSNetworkingPolicy is Enabled and set as appropriate	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates	☐	☐

Appendix: CIS Controls v8 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Enable audit Logs	☐	☐
2.1.2	Ensure audit logs are collected and managed	☐	☐
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
3.2.1	Ensure that the Anonymous Auth is Not Enabled	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	☐	☐
3.2.3	Ensure that a Client CA File is Configured	☐	☐
3.2.4	Ensure that the --read-only-port is disabled	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	☐	☐
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	☐	☐
3.2.7	Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture	☐	☐
3.2.8	Ensure that the --rotate-certificates argument is not present or is set to true	☐	☐
3.2.9	Ensure that the RotateKubeletServerCertificate argument is set to true	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Minimize access to create pods	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.1.5	Ensure that default service accounts are not actively used.	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.1.7	Cluster Access Manager API to streamline and enhance the management of access controls within EKS clusters	☐	☐
4.1.8	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
4.1.9	Minimize access to create persistent volumes	☐	☐
4.1.10	Minimize access to the proxy sub-resource of nodes	☐	☐
4.1.11	Minimize access to webhook configuration objects	☐	☐
4.1.12	Minimize access to the service account token creation	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	☐	☐
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	☐	☐
4.3.1	Ensure CNI plugin supports network policies.	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.4.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.4.2	Consider external secret storage	☐	☐
4.5.1	Create administrative boundaries between resources using namespaces	☐	☐
4.5.2	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning using Amazon ECR image scanning or a third party provider	☐	☐
5.1.2	Minimize user access to Amazon ECR	☐	☐
5.1.3	Minimize cluster access to read-only for Amazon ECR	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.2.1	Prefer using dedicated EKS Service Accounts	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted using Customer Master Keys (CMKs) managed in AWS KMS	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.2	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.4.3	Ensure clusters are created with Private Nodes	☐	☐
5.4.4	Ensure AmazonEKSNetworkingPolicy is Enabled and set as appropriate	☐	☐
5.4.5	Encrypt traffic to HTTPS load balancers with TLS certificates	☐	☐
5.5.1	Manage Kubernetes RBAC users with AWS IAM Authenticator for Kubernetes or Upgrade to AWS CLI v1.16.156 or greater	☐	☐

Appendix: CIS Controls v8 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
	No unmapped recommendations to CIS Controls v8	☐	☐

Appendix: Change History

Date	Version	Changes for this version
10/14/2025	1.8.0	AAC updated and tested against Latest Cluster available v1.34
10/12/2025	1.8.0	AAC testing on versions 1.32,1.33, 1.34
10/12/2025	1.8.0	Benchmark updated to v1.32, 1.33 & 1.34
9/17/2025	1.8.0	Added recommendation 4.1.9 “Minimize access to create persistent volumes”
9/17/2025	1.8.0	Added recommendation 4.1.10 “ Minimize access to the proxy sub-resource of nodes
9/17/2025	1.8.0	Added recommendation 4.1.11 “Minimize access to webhook configuration objects”
9/17/2025	1.8.0	Added recommendation 4.1.12 “Minimize access to the service account token creation”
9/17/2025	1.8.0	Modified recommendation 4.1.6 to align with Kubernetes Benchmark v1.13
9/16/2025	1.8.0	Modified recommendation 4.1.1 to align with Kubernetes Benchmark v1.13
9/1/2025	1.8.0	Modified recommendation 4.1.2 to align with Kubernetes Benchmark v1.13
9/1/2025	1.8.0	Modified recommendation 4.1.3 to align with Kubernetes Benchmark v1.13
9/1/2025	1.8.0	Modified recommendation 4.1.4 to align with Kubernetes Benchmark v1.13
9/1/2025	1.8.0	Modified recommendation 4.1.5 to align with Kubernetes Benchmark v1.13
5/1/2025	1.7.0	AAC updated and tested against Latest Cluster available v1.32
5/1/2025	1.7.0	AAC testing on versions 1.30,1.31, 1.32
4/14/2025	1.7.0	3.1.3 AAC test script edited
4/14/2025	1.7.0	3.1.1 AAC test script edited.

Date	Version	Changes for this version
9/5/2024	1.6.0	AAC updated and tested against Latest Cluster available v1.31
9/5/2024	1.6.0	AAC testing on versions 1.29, 1.30,1.31
9/4/2024	1.6.0	4.5.2 no longer required
9/4/2024	1.6.0	4.1.7 Edited recommendation - changed from "Avoid use of system:masters group"
5/14/2024	1.5.0	AAC updated and tested against Latest Cluster available v1.30
5/12/2024	1.5.0	4.1.7 Cluster Access Manager API to streamline and enhance the management of access controls within EKS clusters
5/12/2024	1.5.0	3.1.1 Ensure that the kubeconfig file permissions are set to 644 or more restrictive
5/12/2024	1.5.0	3.1.2 Ensure that the kubelet kubeconfig file ownership is set to root:root
5/12/2024	1.5.0	3.1.3 Ensure that the kubelet configuration file has permissions set to 644 or more restrictive
5/12/2024	1.5.0	3.1.4 Ensure that the kubelet configuration file ownership is set to root:root
5/12/2024	1.5.0	4.2.1 Minimize the admission of privileged containers
5/12/2024	1.5.0	4.2.2 Minimize the admission of containers wishing to share the host process ID namespace
5/12/2024	1.5.0	4.2.3 Minimize the admission of containers wishing to share the host IPC namespace
5/12/2024	1.5.0	4.2.4 Minimize the admission of containers wishing to share the host network namespace

Date	Version	Changes for this version
5/12/2024	1.5.0	4.2.5 Minimize the admission of containers with allowPrivilegeEscalation
5/12/2024	1.5.0	4.5.3 The default namespace should not be used
5/12/2024	1.5.0	3.3.1 Prefer using a container-optimized OS when possible
5/12/2024	1.5.0	4.2.6 Minimize the admission of root containers
5/12/2024	1.5.0	4.2.7 Minimize the admission of containers with added capabilities
5/12/2024	1.5.0	4.2.8 Minimize the admission of containers with capabilities assigned
5/12/2024	1.5.0	6.1 Consider Fargate for running untrusted workloads
5/12/2024	1.5.0	3.2.4 Ensure that the --read-only-port is disabled
5/12/2024	1.5.0	3.2.8 Ensure that the --rotate-certificates argument is not present or is set to true
5/12/2024	1.5.0	4.1.1 Ensure that the cluster-admin role is only used where required
5/11/2024	1.5.0	4.1.2 Minimize access to secrets
5/11/2024	1.5.0	4.1.4 Minimize access to create pods
5/11/2024	1.5.0	4.1.5 Ensure that default service accounts are not actively used
5/11/2024	1.5.0	4.1.6 Ensure that Service Account Tokens are only mounted where necessary
5/1/2024	1.5.0	Added AAC for 4.1.7 Avoid use of system:masters group

Date	Version	Changes for this version
5/1/2024	1.5.0	Added AAC for 4.3.2 Ensure that all Namespaces have Network Policies defined
5/1/2024	1.5.0	Added AAC for 4.4.1 Prefer using secrets as files over secrets as environment variables
5/1/2024	1.5.0	Added AAC for 5.2.1 Prefer using dedicated EKS Service Accounts
9/17/2023	1.4.0	Added support the latest Kubernetes version Cluster creation option/s
9/15/2023	1.4.0	Ticket 19589 – Reassessed EKS 3.2.7 Hostname override recommendation
5/5/2023	1.3.0	Ticket 18577 – Added specific audit command for ensuring clusters are created with private endpoints – recommendation 5.3.3
4/30/2023	1.3.0	Ticket 18578 – updated audit and remediation guidance for ensure network policy is enabled – recommendation 5.4.4
4/29/2023	1.3.0	Ticket 18576 – updated audit procedure guidance for restricting access to the control plane – recommendation 5.4.1
3/15/2023	1.3.0	Ticket 18580 – Added Audit procedure for running untrusted workloads – recommendation 5.6.1
3/11/2023	1.3.0	Ticket 18579 – Added specific audit and remediation guidance to recommendation 5.4.5
03/11/2023	1.3.0	Ticket 17212 – Edited 4.5 Admission Controls Section. It will be revised in VNext.
02/22/2023	1.3.0	Ticket 17102 – Replaced deprecated and out of date Pod Security Policy verbiage with pod Security Standards.
11/14/2022	1.2.0	Ticket 15928 – Recommendation 3.2.9 updated
11/14/2022	1.2.0	Ticket 15915 – Pod Security Policy deprecated

Date	Version	Changes for this version
11/12/2022	1.2.0	Ticket 15913 – recommendation 3.2.9 updated Event QPS deprecated
11/12/2022	1.2.0	Ticket 16516 – recommendation 3.2.2 updated configuration guidance regarding “always allow”
11/12/2022	1.2.0	Ticket 15485 – ELS Audit Log impact statement updated.
11/12/2022	1.2.0	Ticket 15654 – recommendation 3.2.3 audit
11/01/2022	1.2.0	Ticket 15618 – recommendation 3.2.9 profile updated to level 2
11/01/2022	1.2.0	Ticket 15616 – recommendation 3.2.8 - hostname-overide remediation process updated.
11/01/2022	1.2.0	Ticket 15805 – recommendation 3.3.1 audit procedure updated.
10/10/2022	1.2.0	Ticket 15843 – remediation process updated to set .spec.privileged field to false.
10/10/2022	1.2.0	Ticket 15849 – PSP has allowedCapabilities Set by Default
10/10/2022	1.2.0	Ticket 15850 – recommendation 4.2.7 combined with 4.2.9
10/08/2022	1.2.0	Ticket 15532 – recommendation 5.4.2 updated remediation process
10/08/2022	1.2.0	Ticket 15804 – recommendation 3.3.1 description updated
10/08/2022	1.2.0	Ticket 15602 – recommendation 3.2.6 audit procedure updated
10/08/2022	1.2.0	Ticket 16433 – recommendation 3.2.10 updated the default value
10/08/2022	1.2.0	Ticket 15613 – recommendations 3.2.1, 3.2.2, 3.2.3 updated to reflect authentication vs authorization exec arguments
10/08/2022	1.2.0	Ticket 15677 – recommendation 5.2.1 updated to reflect EKS specifics

Date	Version	Changes for this version
10/08/2022	1.2.0	Ticket 15533 -recommendation 5.4.4 updated recommendation for EKS specifics
9/01/2022	1.2.0	Ticket 15615 – recommendation 3.2.8 updated impact statement
9/01/2022	1.2.0	Ticket 15611 – recommendation 3.2.5 updated audit and remediation methods
9/01/2022	1.2.0	Ticket 15610 – recommendation 3.2.6 updated description
9/01/2022	1.2.0	Ticket 15531 – recommendation 5.4.1 updated remediation process
9/01/2022	1.2.0	Ticket 15533 – recommendation 5.4.4 updated
9/01/2022	1.2.0	Ticket 15599 – recommendation 5.4.5 reviewed load balancer scope
9/01/2022	1.2.0	Ticket 15554 – recommendation 5.5.1 updated default configuration
9/01/2022	1.2.0	Ticket 16573 - recommendation 3.2.1 and 3.2.2 updated and aligned based on new functionality
9/01/2022	1.2.0	Ticket 16612 – recommendation 5.4.2 updated audit procedure
9/01/2022	1.2.0	Ticket 16674 – recommendation 3.2.3 updated out of date guidance
9/01/2022	1.2.0	Ticket 15533 – recommendation 5.5.1 updated description
9/01/2022	1.2.0	Ticket 15539 – recommendation 5.4.3 updated remediation process
9/01/2022	1.2.0	Ticket 15614 – recommendation 3.2.1 updated remediation process
9/01/2022	1.2.0	Ticket 17054 – recommendation 5.4.1 updated for EKS specifically