CIS Azure Kubernetes Service (AKS) Benchmark

v1.5.0 - 04-22-2024

Terms of Use

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

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

Table of Contents

Terms of Use 1
Table of Contents 2
Overview 5
Intended Audience 5
Consensus Guidance 6
Typographical Conventions 7
Recommendation Definitions 8
Title 8
Assessment Status 8
Automated 8
Manual 8
Profile 8
Description 8
Rationale Statement 8
Impact Statement 9
Audit Procedure 9
Remediation Procedure 9
Default Value 9
References 9
CIS Critical Security Controls® (CIS Controls®) 9
Additional Information 9
Profile Definitions 10
Acknowledgements 11
Recommendations 12
1 Master (Control Plane) Components 12
2 Master (Control Plane) Configuration 12
2.1 Logging 13
2.1.1 Enable audit Logs (Manual) 14
3 Worker Nodes 16
3.1 Worker Node Configuration Files 17
3.1.1 Ensure that the kubeconfig file permissions are set to 644 or more restrictive (Automated) 18
3.1.2 Ensure that the kubelet kubeconfig file ownership is set to root:root (Automated) 21
3.1.3 Ensure that the azure.json file has permissions set to 644 or more restrictive (Automated) 24
3.1.4 Ensure that the azure.json file ownership is set to root:root (Automated) 27

3.2 Kubelet **30**
3.2.1 Ensure that the --anonymous-auth argument is set to false (Automated) 31
3.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated) 34
3.2.3 Ensure that the --client-ca-file argument is set as appropriate (Automated) 37
3.2.4 Ensure that the --read-only-port is secured (Automated) 40
3.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated) 42
3.2.6 Ensure that the --make-iptables-util-chains argument is set to true (Automated) 45
3.2.7 Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture (Automated) 48
3.2.8 Ensure that the --rotate-certificates argument is not set to false (Automated) 51
3.2.9 Ensure that the RotateKubeletServerCertificate argument is set to true (Automated) 53

4 Policies **55**
4.1 RBAC and Service Accounts **56**
4.1.1 Ensure that the cluster-admin role is only used where required (Automated) 57
4.1.2 Minimize access to secrets (Automated) 59
4.1.3 Minimize wildcard use in Roles and ClusterRoles (Automated) 61
4.1.4 Minimize access to create pods (Automated) 63
4.1.5 Ensure that default service accounts are not actively used (Automated) 65
4.1.6 Ensure that Service Account Tokens are only mounted where necessary (Automated) 67
4.2 Pod Security Standards **69**
4.2.1 Minimize the admission of privileged containers (Automated) 70
4.2.2 Minimize the admission of containers wishing to share the host process ID namespace (Automated) 73
4.2.3 Minimize the admission of containers wishing to share the host IPC namespace (Automated) 75
4.2.4 Minimize the admission of containers wishing to share the host network namespace (Automated) 77
4.2.5 Minimize the admission of containers with allowPrivilegeEscalation (Automated) 79
4.3 Azure Policy / OPA **81**
4.4 CNI Plugin **82**
4.4.1 Ensure latest CNI version is used (Automated) 83
4.4.2 Ensure that all Namespaces have Network Policies defined (Automated) 85
4.5 Secrets Management **87**
4.5.1 Prefer using secrets as files over secrets as environment variables (Automated) 88
4.5.2 Consider external secret storage (Manual) 90
4.6 General Policies **92**
4.6.1 Create administrative boundaries between resources using namespaces (Manual) 93
4.6.2 Apply Security Context to Your Pods and Containers (Manual) 95
4.6.3 The default namespace should not be used (Automated) 98

5 Managed services **99**
5.1 Image Registry and Image Scanning **100**
5.1.1 Ensure Image Vulnerability Scanning using Azure Defender image scanning or a third party provider (Automated) 101
5.1.2 Minimize user access to Azure Container Registry (ACR) (Manual) 103
5.1.3 Minimize cluster access to read-only for Azure Container Registry (ACR) (Manual) 105
5.1.4 Minimize Container Registries to only those approved (Manual) 106
5.2 Access and identity options for Azure Kubernetes Service (AKS) **108**
5.2.1 Prefer using dedicated AKS Service Accounts (Manual) 109
5.3 Key Management Service (KMS) **111**
5.3.1 Ensure Kubernetes Secrets are encrypted (Manual) 112
5.4 Cluster Networking **113**
5.4.1 Restrict Access to the Control Plane Endpoint (Automated) 114
5.4.2 Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Automated) 117

5.4.3 Ensure clusters are created with Private Nodes (Automated) 119
5.4.4 Ensure Network Policy is Enabled and set as appropriate (Automated) 121
5.4.5 Encrypt traffic to HTTPS load balancers with TLS certificates (Manual) 123
5.5 Authentication and Authorization 124
5.5.1 Manage Kubernetes RBAC users with Azure AD (Manual) 125
5.5.2 Use Azure RBAC for Kubernetes Authorization (Manual) 126
Appendix: Summary Table 128
Appendix: CIS Controls v7 IG 1 Mapped Recommendations 133
Appendix: CIS Controls v7 IG 2 Mapped Recommendations 134
Appendix: CIS Controls v7 IG 3 Mapped Recommendations 136
Appendix: CIS Controls v7 Unmapped Recommendations 138
Appendix: CIS Controls v8 IG 1 Mapped Recommendations 139
Appendix: CIS Controls v8 IG 2 Mapped Recommendations 141
Appendix: CIS Controls v8 IG 3 Mapped Recommendations 143
Appendix: CIS Controls v8 Unmapped Recommendations 145
Appendix: Change History 146

Overview

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

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

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

This document provides prescriptive guidance for running Azure Kubernetes Service (AKS) following recommended security controls. This benchmark only includes controls which can be modified by an end user of Azure AKS and is designed to supersede all previous version of the Azure Kubernetes Service (AKS) Benchmark. It addresses and has been tested against Kubernetes version/s 1.30, 1.29 and 1.28.
As a rule, this benchmark will address the Kubernetes versions available for cluster creation at the beginning of the consensus release period and will address Kubernetes versions that become available after that date in the next release.
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 Azure Kubernetes Service (AKS) using managed and self-managed nodes.

Consensus Guidance

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

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

Typographical Conventions

The following typographical conventions are used throughout this guide:

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

Recommendation Definitions

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

Title

Concise description for the recommendation's intended configuration.

Assessment Status

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

Automated

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

Manual

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

Profile

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

Description

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

Rationale Statement

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

Impact Statement

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

Audit Procedure

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

Remediation Procedure

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

Default Value

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

References

Additional documentation relative to the recommendation.

CIS Critical Security Controls® (CIS Controls®)

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

Additional Information

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

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

  Level 1 profile for running Automated Assessment

- **Level 2**

  Level 2 profile for running Automated Assessment

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:

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.

Special Thanks to:

Author
Randall Mowen

Editors
Mark Larinde and Randall Mowen

Contributors
Paavan Mistry
Rafael Pereyra
Angus Lees
Abeer Sethi
Gert Van Den Berg
Rory MCcune
Thomas Dupas
Corey McDonald

Recommendations

1 Master (Control Plane) Components

Security is a shared responsibility between Microsoft and the Azure (AKS) customer.

Master security In AKS, the Kubernetes master components are part of the managed service provided by Microsoft. Each AKS cluster has its own single-tenanted, dedicated Kubernetes master to provide the API Server, Scheduler, etc. This master is managed and maintained by Microsoft.

By default, the Kubernetes API server uses a public IP address and a fully qualified domain name (FQDN). You can limit access to the API server endpoint using authorized IP ranges. You can also create a fully private cluster to limit API server access to your virtual network.

You can control access to the API server using Kubernetes role-based access control (Kubernetes RBAC) and Azure RBAC. For more information, see Azure AD integration with AKS.

2 Master (Control Plane) Configuration

With Azure Kubernetes Service (AKS), the master components such as the kube-apiserver and kube-controller-manager are provided as a managed service. You create and manage the nodes that run the kubelet and container runtime, and deploy your applications through the managed Kubernetes API server. To help troubleshoot your application and services, you may need to view the logs generated by these master components. This section shows you how to use Azure Monitor logs to enable and query the logs from the Kubernetes master (control plane) components.

2.1 Logging

2.1.1 Enable audit Logs (Manual)

Profile Applicability:

- Level 1

Description:

With Azure Kubernetes Service (AKS), the control plane components such as the kube-apiserver and kube-controller-manager are provided as a managed service. You create and manage the nodes that run the kubelet and container runtime, and deploy your applications through the managed Kubernetes API server. To help troubleshoot your application and services, you may need to view the logs generated by these control plane components.

To help collect and review data from multiple sources, Azure Monitor logs provides a query language and analytics engine that provides insights to your environment. A workspace is used to collate and analyze the data, and can integrate with other Azure services such as Application Insights and Security Center.

Rationale:

Exporting logs and metrics to a dedicated, persistent datastore ensures availability of audit data following a cluster security event, and provides a central location for analysis of log and metric data collated from multiple sources.

Impact:

What is collected from Kubernetes clusters Container insights includes a predefined set of metrics and inventory items collected that are written as log data in your Log Analytics workspace. All metrics listed below are collected by default every one minute.

Node metrics collected The following list is the 24 metrics per node that are collected:

cpuUsageNanoCores cpuCapacityNanoCores cpuAllocatableNanoCores memoryRssBytes memoryWorkingSetBytes memoryCapacityBytes memoryAllocatableBytes restartTimeEpoch used (disk) free (disk) used_percent (disk) io_time (diskio) writes (diskio) reads (diskio) write_bytes (diskio) write_time (diskio) iops_in_progress (diskio) read_bytes (diskio) read_time (diskio) err_in (net) err_out (net) bytes_recv (net) bytes_sent (net) Kubelet_docker_operations (kubelet) Container metrics The following list is the eight metrics per container collected:

cpuUsageNanoCores cpuRequestNanoCores cpuLimitNanoCores memoryRssBytes memoryWorkingSetBytes memoryRequestBytes memoryLimitBytes restartTimeEpoch Cluster inventory The following list is the cluster inventory data collected by default:

KubePodInventory – 1 per minute per container KubeNodeInventory – 1 per node per minute KubeServices – 1 per service per minute ContainerInventory – 1 per container per minute

Audit:

Remediation:

Azure audit logs are enabled and managed in the Azure portal. To enable log collection for the Kubernetes master components in your AKS cluster, open the Azure portal in a web browser and complete the following steps:

1. Select the resource group for your AKS cluster, such as myResourceGroup. Don't select the resource group that contains your individual AKS cluster resources, such as MC_myResourceGroup_myAKSCluster_eastus.
2. On the left-hand side, choose Diagnostic settings.
3. Select your AKS cluster, such as myAKSCluster, then choose to Add diagnostic setting.
4. Enter a name, such as myAKSClusterLogs, then select the option to Send to Log Analytics.
5. Select an existing workspace or create a new one. If you create a workspace, provide a workspace name, a resource group, and a location.
6. In the list of available logs, select the logs you wish to enable. For this example, enable the kube-audit and kube-audit-admin logs. Common logs include the kube-apiserver, kube-controller-manager, and kube-scheduler. You can return and change the collected logs once Log Analytics workspaces are enabled.
7. When ready, select Save to enable collection of the selected logs.

Default Value:

By default, cluster control plane logs aren't sent to be Logged.

References:

1. https://kubernetes.io/docs/tasks/debug-application-cluster/audit/
2. https://docs.microsoft.com/en-us/azure/aks/view-master-logs
3. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-logging-threat-detection#lt-4-enable-logging-for-azure-resources

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.	●	●	●

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.	●	●	●
v7	6 Maintenance, Monitoring and Analysis of Audit Logs Maintenance, Monitoring and Analysis of Audit Logs			

3 Worker Nodes

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

3.1 Worker Node Configuration Files

Node security AKS nodes are Azure virtual machines that you manage and maintain. Linux nodes run an optimized Ubuntu distribution using the Moby container runtime. Windows Server nodes run an optimized Windows Server 2019 release and also use the Moby container runtime. When an AKS cluster is created or scaled up, the nodes are automatically deployed with the latest OS security updates and configurations.

The Azure platform automatically applies OS security patches to Linux nodes on a nightly basis. If a Linux OS security update requires a host reboot, that reboot is not automatically performed. You can manually reboot the Linux nodes, or a common approach is to use Kured, an open-source reboot daemon for Kubernetes. Kured runs as a DaemonSet and monitors each node for the presence of a file indicating that a reboot is required. Reboots are managed across the cluster using the same cordon and drain process as a cluster upgrade.

For Windows Server nodes, Windows Update does not automatically run and apply the latest updates. On a regular schedule around the Windows Update release cycle and your own validation process, you should perform an upgrade on the Windows Server node pool(s) in your AKS cluster. This upgrade process creates nodes that run the latest Windows Server image and patches, then removes the older nodes. For more information on this process, see Upgrade a node pool in AKS.

Nodes are deployed into a private virtual network subnet, with no public IP addresses assigned. For troubleshooting and management purposes, SSH is enabled by default. This SSH access is only available using the internal IP address.

To provide storage, the nodes use Azure Managed Disks. For most VM node sizes, these are Premium disks backed by high-performance SSDs. The data stored on managed disks is automatically encrypted at rest within the Azure platform. To improve redundancy, these disks are also securely replicated within the Azure datacenter.

Kubernetes environments, in AKS or elsewhere, currently aren't completely safe for hostile multi-tenant usage. Additional security features like Pod Security Policies, or more fine-grained Kubernetes role-based access control (Kubernetes RBAC) for nodes, make exploits more difficult. However, for true security when running hostile multi-tenant workloads, a hypervisor is the only level of security that you should trust. The security domain for Kubernetes becomes the entire cluster, not an individual node. For these types of hostile multi-tenant workloads, you should use physically isolated clusters. For more information on ways to isolate workloads, see Best practices for cluster isolation in AKS.

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:

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 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 Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kube-proxy/
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-3-establish-secure-configurations-for-compute-resources

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 Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kube-proxy/
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-3-establish-secure-configurations-for-compute-resources

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 azure.json file has permissions set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

The azure.json file in an Azure Kubernetes Service (AKS) cluster is a configuration file used by the Kubernetes cloud provider integration for Azure. This file contains essential details that allow the Kubernetes cluster to interact with Azure resources effectively. It's part of the Azure Cloud Provider configuration, enabling Kubernetes components to communicate with Azure services for features like load balancers, storage, and networking.

Ensure the file has permissions of 644 or more restrictive.

Rationale:

The azure.json file in AKS structure typically includes:

- Tenant ID: The Azure Tenant ID where the AKS cluster resides.
- Subscription ID: The Azure Subscription ID used for billing and resource management.
- AAD Client ID: The Azure Active Directory (AAD) application client ID used by the Kubernetes cloud provider to interact with Azure resources.
- AAD Client Secret: The secret for the AAD application.
- Resource Group: The name of the resource group where the AKS cluster resources are located.
- Location: The Azure region where the AKS cluster is deployed.
- VM Type: Specifies the type of VMs used by the cluster (e.g., standard VMs or Virtual Machine Scale Sets).
- Subnet Name, Security Group Name, Vnet Name, and Vnet Resource Group: Networking details for the cluster.
- Route Table Name: The name of the route table for the cluster.
- Storage Account Type: The default type of storage account to use for Kubernetes persistent volumes.

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/azure.json` which is the location of the Kubelet config file.
Run the following command:

```
stat -c %a /etc/kubernetes/azure.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/azure.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/azure.json
```

Default Value:

See the Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-3-establish-secure-configurations-for-compute-resources

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 azure.json file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

The azure.json file in an Azure Kubernetes Service (AKS) cluster is a configuration file used by the Kubernetes cloud provider integration for Azure. This file contains essential details that allow the Kubernetes cluster to interact with Azure resources effectively. It's part of the Azure Cloud Provider configuration, enabling Kubernetes components to communicate with Azure services for features like load balancers, storage, and networking.

Ensure that the file is owned by root:root.

Rationale:

The azure.json file in AKS structure typically includes:

- Tenant ID: The Azure Tenant ID where the AKS cluster resides.
- Subscription ID: The Azure Subscription ID used for billing and resource management.
- AAD Client ID: The Azure Active Directory (AAD) application client ID used by the Kubernetes cloud provider to interact with Azure resources.
- AAD Client Secret: The secret for the AAD application.
- Resource Group: The name of the resource group where the AKS cluster resources are located.
- Location: The Azure region where the AKS cluster is deployed.
- VM Type: Specifies the type of VMs used by the cluster (e.g., standard VMs or Virtual Machine Scale Sets).
- Subnet Name, Security Group Name, Vnet Name, and Vnet Resource Group: Networking details for the cluster.
- Route Table Name: The name of the route table for the cluster.
- Storage Account Type: The default type of storage account to use for Kubernetes persistent volumes.

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/azure.json` which is the location of the Kubelet config file.
Run the following command:

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

Default Value:

See the Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kube-proxy/
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-3-establish-secure-configurations-for-compute-resources

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

This section contains recommendations for kubelet configuration.

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

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

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

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

3.2.1 Ensure that the --anonymous-auth argument is set to false (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:

Anonymous requests will be rejected.

Audit:

Audit Method 1:
If using a Kubelet configuration file, check that there is an entry for `authentication: anonymous: enabled` set to `false`.
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:

```
sudo more /etc/kubernetes/kubelet/kubelet-config.json
```

Verify that the `"authentication": { "anonymous": { "enabled": false }` argument is set to `false`.
Audit Method 2:
If using the api configz endpoint consider searching for the status of `authentication... "anonymous":{"enabled":false}` 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.aks.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 false

```
"anonymous": "enabled": false
```

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.

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

Remediation Method 3:
If using the api configz endpoint consider searching for the status of `"authentication.*anonymous":{"enabled":false}"` 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.aks.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 Azure AKS documentation for the default value.

References:

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

2. https://kubernetes.io/docs/admin/kubelet-authentication-authorization/#kubelet-authentication
3. https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/
4. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-governance-strategy#gs-6-define-identity-and-privileged-access-strategy

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6 Access Control Management Use processes and tools to create, assign, manage, and revoke access credentials and privileges for user, administrator, and service accounts for enterprise assets and software.			
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

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

Impact:

Unauthorized requests will be denied.

Audit:

Audit Method 1:
If using a Kubelet configuration file, check that there is an entry for `"authentication": "webhook": "enabled"` set to `true`.
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:

```
sudo more /etc/kubernetes/kubelet/kubelet-config.json
```

Verify that the `"authentication": {"webhook": { "enabled": is set to true.`
If the `"authentication": {"mode": {` argument is present check that it is not set to `AlwaysAllow`. If it is not present check that there is a Kubelet config file specified by `--config`, and that file sets `"authentication": {"mode": {` to something other than `AlwaysAllow`.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `authentication... "webhook":{"enabled":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.aks.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 false

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

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.

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

Remediation Method 3:

If using the api configz endpoint consider searching for the status of `"authentication.*webhook":{"enabled":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.aks.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 Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/admin/kubelet-authentication-authorization/#kubelet-authentication
3. https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/
4. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-governance-strategy#gs-6-define-identity-and-privileged-access-strategy

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6 Access Control Management Use processes and tools to create, assign, manage, and revoke access credentials and privileges for user, administrator, and service accounts for enterprise assets and software.			
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

3.2.3 Ensure that the --client-ca-file argument is set as appropriate (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:
If using a Kubelet configuration file, check that there is an entry for `"x509": {"clientCAFile:"` set to the location of the client certificate authority file.
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:

```
sudo more /etc/kubernetes/kubelet/kubelet-config.json
```

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

Audit Method 2:

If using the api configz endpoint consider searching for the status of `authentication.. x509":("clientCAFile":"/etc/kubernetes/pki/ca.crt` 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.aks.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 false

```
"authentication": { "x509": {"clientCAFile:" to the location of the client CA
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.

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

Remediation Method 3:

If using the api configz endpoint consider searching for the status of `"authentication.*x509":("clientCAFile":"/etc/kubernetes/pki/ca.crt"` 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.aks.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 Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/
3. https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/
4. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-data-protection#dp-4-encrypt-sensitive-information-in-transit

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6 Access Control Management Use processes and tools to create, assign, manage, and revoke access credentials and privileges for user, administrator, and service accounts for enterprise assets and software.			
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 secured (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 false

```
readOnlyPort to 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 all 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 Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-3-establish-secure-configurations-for-compute-resources

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	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.aks.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.aks.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 Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://github.com/kubernetes/kubernetes/pull/18552
3. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-3-establish-secure-configurations-for-compute-resources

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

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:
If using a Kubelet configuration file, check that there is an entry for `makeIPTablesUtilChains` set to `true`.
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 if the `makeIPTablesUtilChains` argument exists then it is set to `true`.
If the `--make-iptables-util-chains` argument does not exist, and there is a Kubelet config file specified by `--config`, verify that the file does not set `makeIPTablesUtilChains` to `false`.

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.aks.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 false

```
"makeIPTablesUtilChains": true
```

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.aks.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 Azure AKS documentation for the default value.

References:

1. https://kubernetes.io/docs/admin/kubelet/
2. https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/
3. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-network-security#ns-1-implement-security-for-internal-traffic

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

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

Profile Applicability:

- Level 2

Description:

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

Rationale:

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

Impact:

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

Audit:

Audit Method 1:
First, SSH to each node.
Run the following command on each node to find the Kubelet process:

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

In the output of the above command review the value set for the `--eventRecordQPS` argument and determine whether this has been set to an appropriate level for the cluster. The value of `0` can be used to ensure that all events are captured.
If the `--eventRecordQPS` argument does not exist, check that there is a Kubelet config file specified by `--config` and review the value in this location.
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
```

If there is an entry for `eventRecordQPS` check that it is set to 0 or an appropriate level for the cluster.

Audit Method 2:

If using the api configz endpoint consider searching for the status of `eventRecordQPS` 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.aks.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 5 or a value greater or equal to 0

```
"eventRecordQPS": 5
```

Check that `/etc/systemd/system/kubelet.service.d/10-kubelet-args.conf` does not define an executable argument for `eventRecordQPS` because this would override your Kubelet config.

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.

```
--eventRecordQPS=5
```

Remediation Method 3:

If using the api configz endpoint consider searching for the status of `"eventRecordQPS"` 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.aks.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 AKS 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/
4. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-logging-threat-detection

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

3.2.8 Ensure that the --rotate-certificates argument is not set to false (Automated)

Profile Applicability:

- Level 2

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 AKS 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/
6. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-data-protection#dp-4-encrypt-sensitive-information-in-transit

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 security triad.

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

Impact:

None

Audit:

Audit Method 1:
If using a Kubelet configuration file, check that there is an entry for `RotateKubeletServerCertificate` is set to `true`.
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 `RotateKubeletServerCertificate` argument exists 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.aks.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

```
"RotateKubeletServerCertificate":true
```

Remediation Method 2:

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

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.aks.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 AKS 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 Azure AKS customer environment.

4.1 RBAC and Service Accounts

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

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
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-privileged-access#pa-7-follow-just-enough-administration-least-privilege-principle

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	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

4.1.2 Minimize access to secrets (Automated)

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
```

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-identity-management#im-7-eliminate-unintended-credential-exposure

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 (Automated)

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.

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-privileged-access#pa-7-follow-just-enough-administration-least-privilege-principle

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	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

4.1.4 Minimize access to create pods (Automated)

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
```

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-privileged-access#pa-7-follow-just-enough-administration-least-privilege-principle

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.		●	●

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

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://docs.microsoft.com/security/benchmark/azure/security-controls-v2-identity-management#im-2-manage-application-identities-securely-and-automatically

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.7 Manage Default Accounts on Enterprise Assets and Software Manage default accounts on enterprise assets and software, such as root, administrator, and other pre-configured vendor accounts. Example implementations can include: disabling default accounts or making them unusable.	●	●	●
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	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.		●	●

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

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.
Set SERVICE_ACCOUNT and POD variables to appropriate values

```
automountServiceAccountToken: false
```

Remediation:

Modify the definition of pods and service accounts which do not need to mount service account tokens to disable it.

Default Value:

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

References:

1. https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-identity-management#im-2-manage-application-identities-securely-and-automatically

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.		●	●

4.2 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 (Automated)

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 -r '.items[] |
select(.spec.containers[].securityContext.privileged == true) |
select(.metadata.namespace != "kube-system" and .metadata.namespace !=
"gatekeeper-system" and .metadata.namespace != "azure-arc" and
.metadata.namespace != "azure-extensions-usage-system") | "\(.metadata.name)
\(.metadata.namespace)"'
```

When creating a Pod Security Policy, ["kube-system", "gatekeeper-system", "azure-arc", "azure-extensions-usage-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
```

Pod Security Policies and Assignments can be found by searching for Policies in the Azure Portal. A detailed step-by-step guide can be found here: https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes

Default Value:

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

References:

1. https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes
2. https://learn.microsoft.com/en-us/azure/aks/use-azure-policy
3. https://kubernetes.io/docs/concepts/security/pod-security-admission/

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.		●	●

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

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 -r '.items[] |
select(.spec.hostPID == true) | select(.metadata.namespace != "kube-system"
and .metadata.namespace != "gatekeeper-system" and .metadata.namespace !=
"azure-arc" and .metadata.namespace != "azure-extensions-usage-system") |
"\(.metadata.name) \(.metadata.namespace)"'
```

When creating a Pod Security Policy, ["kube-system", "gatekeeper-system", "azure-arc", "azure-extensions-usage-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.
Pod Security Policies and Assignments can be found by searching for Policies in the Azure Portal. A detailed step-by-step guide can be found here: https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes

Default Value:

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

References:

1. https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes
2. https://kubernetes.io/docs/concepts/security/pod-security-admission/

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.		●	●

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

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 -r '.items[] |
select(.spec.hostIPC == true) | select(.metadata.namespace != "kube-system"
and .metadata.namespace != "gatekeeper-system" and .metadata.namespace !=
"azure-arc" and .metadata.namespace != "azure-extensions-usage-system") |
"\(.metadata.name) \(.metadata.namespace)"'
```

When creating a Pod Security Policy, ["kube-system", "gatekeeper-system", "azure-arc", "azure-extensions-usage-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.
Pod Security Policies and Assignments can be found by searching for Policies in the Azure Portal. A detailed step-by-step guide can be found here: https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes

Default Value:

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

References:

1. https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes
2. https://kubernetes.io/docs/concepts/security/pod-security-admission/
3. https://learn.microsoft.com/en-us/azure/aks/use-psa

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.		●	●

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

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 -r '.items[] |
select(.spec.hostNetwork == true) | select(.metadata.namespace != "kube-
system" and .metadata.namespace != "gatekeeper-system" and
.metadata.namespace != "azure-arc" and .metadata.namespace != "azure-
extensions-usage-system") | "\(.metadata.name) \(.metadata.namespace)"'
```

When creating a Pod Security Policy, ["kube-system", "gatekeeper-system", "azure-arc", "azure-extensions-usage-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.
Pod Security Policies and Assignments can be found by searching for Policies in the Azure Portal. A detailed step-by-step guide can be found here: https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes

Default Value:

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

References:

1. https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes
2. https://learn.microsoft.com/en-us/azure/aks/use-azure-policy
3. https://learn.microsoft.com/en-us/azure/aks/use-psa

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.		●	●

4.2.5 Minimize the admission of containers with allowPrivilegeEscalation (Automated)

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 -r '.items[] |
select(any(.spec.containers[]; .securityContext.allowPrivilegeEscalation ==
true)) | select(.metadata.namespace != "kube-system" and .metadata.namespace
!= "gatekeeper-system" and .metadata.namespace != "azure-arc" and
.metadata.namespace != "azure-extensions-usage-system") | "\(.metadata.name)
\(.metadata.namespace)"'
```

When creating a Pod Security Policy, ["kube-system", "gatekeeper-system", "azure-arc", "azure-extensions-usage-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`.
Pod Security Policies and Assignments can be found by searching for Policies in the Azure Portal. A detailed step-by-step guide can be found here: https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes

Default Value:

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

References:

1. https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes
2. https://learn.microsoft.com/en-us/azure/aks/use-azure-policy
3. https://learn.microsoft.com/en-us/azure/aks/use-psa

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.		●	●

4.3 Azure Policy / OPA

A more modern alternative to the PSP is the Open Policy Agent (OPA) and OPA Gatekeeper. OPA is an admission controller which is integrated with the OPA Constraint Framework to enforce Custom Resource Definition (CRD) based policies and allow declaratively configured policies to be reliably shareable. The Kubernetes projects is shifting focus from PSPs to OPAs.

Finally, third party agents such Aqua, Twistlock (Prisma), and Sysdig can offer similar capabilities or manage PSP's themselves.

Azure Policy extends Gatekeeper v3, an admission controller webhook for Open Policy Agent (OPA), to apply at-scale enforcements and safeguards on your clusters in a centralized, consistent manner. Azure Policy makes it possible to manage and report on the compliance state of your Kubernetes clusters from one place. The add-on enacts the following functions:

- Checks with Azure Policy service for policy assignments to the cluster.
- Deploys policy definitions into the cluster as constraint template and constraint custom resources.
- Reports auditing and compliance details back to Azure Policy service.

Azure Policy for Kubernetes supports the following cluster environments:

- Azure Kubernetes Service (AKS)
- Azure Arc enabled Kubernetes
- AKS Engine

4.4 CNI Plugin

4.4.1 Ensure latest CNI version is used (Automated)

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:

Ensure CNI plugin supports network policies.
Set Environment Variables to run

```
export RESOURCE_GROUP=Resource Group Name
export CLUSTER_NAME=Cluster Name
```

Azure command to check for CNI plugin:

```
az aks show --resource-group ${RESOURCE_GROUP} --name ${CLUSTER_NAME} --query
"networkProfile"
```

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://docs.microsoft.com/security/benchmark/azure/security-controls-v2-network-security#ns-1-implement-security-for-internal-traffic

Additional Information:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.5 Use Up-to-Date and Trusted Third-Party Software Components Use up-to-date and trusted third-party software components. When possible, choose established and proven frameworks and libraries that provide adequate security. Acquire these components from trusted sources or evaluate the software for vulnerabilities before use.		●	●
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.		●	●

4.4.2 Ensure that all Namespaces have Network Policies defined (Automated)

Profile Applicability:

- Level 2

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/
4. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-network-security#ns-1-implement-security-for-internal-traffic

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13.4 Perform Traffic Filtering Between Network Segments Perform traffic filtering between network segments, where appropriate.		●	●
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.		●	●

4.5 Secrets Management

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

Profile Applicability:

- Level 2

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
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-identity-management#im-7-eliminate-unintended-credential-exposure

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	6 Access Control Management Use processes and tools to create, assign, manage, and revoke access credentials and privileges for user, administrator, and service accounts for enterprise assets and software.			
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.			●

4.5.2 Consider external secret storage (Manual)

Profile Applicability:

- Level 2

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.

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-identity-management#im-7-eliminate-unintended-credential-exposure

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6 Access Control Management Use processes and tools to create, assign, manage, and revoke access credentials and privileges for user, administrator, and service accounts for enterprise assets and software.			
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.			●

4.6 General Policies

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

4.6.1 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 Azure AKS 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:

When you create an AKS cluster, the following namespaces are available:

NAMESPACES Namespace Description default Where pods and deployments are created by default when none is provided. In smaller environments, you can deploy applications directly into the default namespace without creating additional logical separations. When you interact with the Kubernetes API, such as with kubectl get pods, the default namespace is used when none is specified. kube-system Where core resources exist, such as network features like DNS and proxy, or the Kubernetes dashboard. You typically don't deploy your own applications into this namespace. kube-public Typically not used, but can be used for resources to be visible across the whole cluster, and can be viewed by any user.

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
3. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-governance-strategy#gs-1-define-asset-management-and-data-protection-strategy
4. https://docs.microsoft.com/en-us/azure/aks/concepts-clusters-workloads#:~:text=Kubernetes%20resources%2C%20such%20as%20pods,or%20manage%20access%20to%20resources.&text=When%20you%20interact%20with%20the,used%20when%20none%20is%20specified.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.6 Securely Manage Enterprise Assets and Software Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.	●	●	●
v7	14 Controlled Access Based on the Need to Know Controlled Access Based on the Need to Know			

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

Profile Applicability:

- Level 2

Description:

Apply Security Context to Your Pods and Containers

Rationale:

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

Impact:

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

Audit:

Review the pod definitions in your cluster and verify that you have security contexts defined as appropriate.

Remediation:

As a best practice we recommend that you scope the binding for privileged pods to service accounts within a particular namespace, e.g. kube-system, and limiting access to that namespace. For all other serviceaccounts/namespaces, we recommend implementing a more restrictive policy such as this:

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
    name: restricted
    annotations:
    seccomp.security.alpha.kubernetes.io/allowedProfileNames:
'docker/default,runtime/default'
    apparmor.security.beta.kubernetes.io/allowedProfileNames:
'runtime/default'
    seccomp.security.alpha.kubernetes.io/defaultProfileName:
'runtime/default'
    apparmor.security.beta.kubernetes.io/defaultProfileName:
'runtime/default'
spec:
    privileged: false
    # Required to prevent escalations to root.
    allowPrivilegeEscalation: false
    # This is redundant with non-root + disallow privilege escalation,
    # but we can provide it for defense in depth.
    requiredDropCapabilities:
    - ALL
    # Allow core volume types.
    volumes:
    - 'configMap'
    - 'emptyDir'
    - 'projected'
    - 'secret'
    - 'downwardAPI'
    # Assume that persistentVolumes set up by the cluster admin are safe to
use.
    - 'persistentVolumeClaim'
    hostNetwork: false
    hostIPC: false
    hostPID: false
    runAsUser:
    # Require the container to run without root privileges.
    rule: 'MustRunAsNonRoot'
    seLinux:
    # This policy assumes the nodes are using AppArmor rather than SELinux.
    rule: 'RunAsAny'
    supplementalGroups:
    rule: 'MustRunAs'
    ranges:
        # Forbid adding the root group.
        - min: 1
        max: 65535
    fsGroup:
    rule: 'MustRunAs'
    ranges:
        # Forbid adding the root group.
        - min: 1
        max: 65535
    readOnlyRootFilesystem: false
```

This policy prevents pods from running as privileged or escalating privileges. It also restricts the types of volumes that can be mounted and the root supplemental groups that can be added.
Another, albeit similar, approach is to start with policy that locks everything down and incrementally add exceptions for applications that need looser restrictions such as logging agents which need the ability to mount a host path.

Default Value:

By default, no security contexts are automatically applied to pods.

References:

1. https://kubernetes.io/docs/concepts/policy/security-context/
2. https://learn.cisecurity.org/benchmarks
3. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-3-establish-secure-configurations-for-compute-resources

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.6 Securely Manage Enterprise Assets and Software Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.	●	●	●
v7	5 Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers			

4.6.3 The default namespace should not be used (Automated)

Profile Applicability:

- Level 2

Description:

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

Rationale:

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

Impact:

None

Audit:

Run this command to list objects in default namespace

```
kubectl get all -n default
```

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

Remediation:

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

Default Value:

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

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-3-establish-secure-configurations-for-compute-resources

CIS Controls:

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

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

Profile Applicability:

- Level 1

Description:

Scan images being deployed to Azure (AKS) for vulnerabilities.

Vulnerability scanning for images stored in Azure Container Registry is generally available in Azure Security Center. This capability is powered by Qualys, a leading provider of information security.

When you push an image to Container Registry, Security Center automatically scans it, then checks for known vulnerabilities in packages or dependencies defined in the file.

When the scan completes (after about 10 minutes), Security Center provides details and a security classification for each vulnerability detected, along with guidance on how to remediate issues and protect vulnerable attack surfaces.

Rationale:

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

Impact:

When using an Azure container registry, you might occasionally encounter problems. For example, you might not be able to pull a container image because of an issue with Docker in your local environment. Or, a network issue might prevent you from connecting to the registry.

Audit:

Check Azure Defender Plan for Container Registries: This command shows whether Azure Defender for container registries (part of Azure Security Center/Defender for Cloud) is enabled, which includes the image scanning feature.

```
az security pricing show --name ContainerRegistry
```

The output will indicate whether the plan is enabled. Look for the "pricingTier" property in the output; if it's set to "Standard", then Azure Defender for container registries is enabled. If it's set to "Free", then it's not enabled.

Remediation:

Enable Azure Defender for Container Registries: If you find that Azure Defender for container registries is not enabled and you wish to enable it, you can do so using the following command:

```
az security pricing create --name ContainerRegistry --tier Standard
```

Please note, enabling Azure Defender for container registries incurs additional costs, so be sure to review the pricing details on the official Azure documentation before enabling it.

Default Value:

Images are not scanned by Default.

References:

1. https://docs.microsoft.com/en-us/azure/security-center/defender-for-container-registries-usage
2. https://docs.microsoft.com/en-us/azure/container-registry/container-registry-check-health
3. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-posture-vulnerability-management#pv-6-perform-software-vulnerability-assessments

CIS Controls:

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 Continuous Vulnerability Management Continuous Vulnerability Management			
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 Azure Container Registry (ACR) (Manual)

Profile Applicability:

- Level 1

Description:

Restrict user access to Azure Container Registry (ACR), limiting interaction with build images to only authorized personnel and service accounts.

Rationale:

Weak access control to Azure Container Registry (ACR) may allow malicious users to replace built images with vulnerable containers.

Impact:

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

Audit:

Remediation:

Azure Container Registry
If you use Azure Container Registry (ACR) as your container image store, you need to grant permissions to the service principal for your AKS cluster to read and pull images. Currently, the recommended configuration is to use the az aks create or az aks update command to integrate with a registry and assign the appropriate role for the service principal. For detailed steps, see Authenticate with Azure Container Registry from Azure Kubernetes Service.
To avoid needing an Owner or Azure account administrator role, you can configure a service principal manually or use an existing service principal to authenticate ACR from AKS. For more information, see ACR authentication with service principals or Authenticate from Kubernetes with a pull secret.

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-privileged-access#pa-7-follow-just-enough-administration-least-privilege-principle

CIS Controls:

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

5.1.3 Minimize cluster access to read-only for Azure Container Registry (ACR) (Manual)

Profile Applicability:

- Level 1

Description:

Configure the Cluster Service Account with Storage Object Viewer Role to only allow read-only access to Azure Container Registry (ACR)

Rationale:

The Cluster Service Account does not require administrative access to Azure ACR, only requiring pull access to containers to deploy onto Azure AKS. 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:

Remediation:

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-data-protection#dp-2-protect-sensitive-data

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	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.		●	●

5.1.4 Minimize Container Registries to only those approved (Manual)

Profile Applicability:

- Level 2

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:

If you are using Azure Container Registry you have this option:
https://docs.microsoft.com/en-us/azure/container-registry/container-registry-firewall-access-rules
For other non-AKS repos using admission controllers or Azure Policy will also work.
Limiting or locking down egress traffic is also recommended:
https://docs.microsoft.com/en-us/azure/aks/limit-egress-traffic

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-asset-management#am-6-use-only-approved-applications-in-compute-resources
2. https://docs.microsoft.com/en-us/azure/aks/limit-egress-traffic
3. https://docs.microsoft.com/en-us/azure/container-registry/container-registry-firewall-access-rules

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	5.2 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.		●	●
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.		●	●

5.2 Access and identity options for Azure Kubernetes Service (AKS)

This section contains recommendations relating to access and identity options for Azure (AKS).

5.2.1 Prefer using dedicated AKS Service Accounts (Manual)

Profile Applicability:

- Level 1

Description:

Kubernetes workloads should not use cluster node service accounts to authenticate to Azure AKS APIs. Each Kubernetes workload that needs to authenticate to other Azure Web Services using IAM should be provisioned with a dedicated Service account.

Rationale:

Manual approaches for authenticating Kubernetes workloads running on Azure AKS against Azure 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.

Remediation:

Azure Active Directory integration
The security of AKS clusters can be enhanced with the integration of Azure Active Directory (AD). Built on decades of enterprise identity management, Azure AD is a multi-tenant, cloud-based directory, and identity management service that combines core directory services, application access management, and identity protection. With Azure AD, you can integrate on-premises identities into AKS clusters to provide a single source for account management and security.
Azure Active Directory integration with AKS clusters
With Azure AD-integrated AKS clusters, you can grant users or groups access to Kubernetes resources within a namespace or across the cluster. To obtain a kubectl configuration context, a user can run the az aks get-credentials command. When a user then interacts with the AKS cluster with kubectl, they're prompted to sign in with their Azure AD credentials. This approach provides a single source for user account management and password credentials. The user can only access the resources as defined by the cluster administrator.
Azure AD authentication is provided to AKS clusters with OpenID Connect. OpenID Connect is an identity layer built on top of the OAuth 2.0 protocol. For more information on OpenID Connect, see the Open ID connect documentation. From inside of the Kubernetes cluster, Webhook Token Authentication is used to verify authentication tokens. Webhook token authentication is configured and managed as part of the AKS cluster.

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-identity-management#im-2-manage-application-identities-securely-and-automatically

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5 Account Management Use processes and tools to assign and manage authorization to credentials for user accounts, including administrator accounts, as well as service accounts, to enterprise assets and software.			
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 Key Management Service (KMS)

5.3.1 Ensure Kubernetes Secrets are encrypted (Manual)

Profile Applicability:

- Level 1

Description:

Encryption at Rest is a common security requirement. In Azure, organizations can encrypt data at rest without the risk or cost of a custom key management solution. Organizations have the option of letting Azure completely manage Encryption at Rest. Additionally, organizations have various options to closely manage encryption or encryption keys.

Rationale:

Audit:

Remediation:

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-data-protection#dp-5-encrypt-sensitive-data-at-rest

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 Azure (AKS).

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 Azure virtual machines 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.

Limitations IP authorized ranges can't be applied to the private api server endpoint, they only apply to the public API server Availability Zones are currently supported for certain regions. Azure Private Link service limitations apply to private clusters. No support for Azure DevOps Microsoft-hosted Agents with private clusters. Consider to use Self-hosted Agents. For customers that need to enable Azure Container Registry to work with private AKS, the Container Registry virtual network must be peered with the agent cluster virtual network.

Audit:

Check for the following to be 'enabled: true'

```
export CLUSTER_NAME=<your cluster name>
export RESOURCE_GROUP=<your resource group name>
az aks show --name ${CLUSTER_NAME} --resource-group ${RESOURCE_GROUP} --query
"apiServerAccessProfile.enablePublicFqdn"
```

This command queries for the enablePublicFqdn property within the apiServerAccessProfile of your AKS cluster. The output will be true if endpointPublicAccess is enabled, allowing access to the AKS cluster API server from the internet. If it's false, endpointPublicAccess is disabled, meaning the API server is not accessible over the internet, which is a common configuration for private clusters.

```
az aks show --name ${CLUSTER_NAME} --resource-group ${RESOURCE_GROUP} --query
"apiServerAccessProfile.enablePrivateCluster"
```

This command queries the enablePrivateCluster property within the apiServerAccessProfile of your AKS cluster. If the output is true, it indicates that endpointPrivateAccess is enabled, and the AKS cluster API server is configured to be accessible only via a private endpoint. If the output is false, the cluster is not configured for private access only, and the API server might be accessible over the internet depending on other settings.
Check for the following is not null:

```
az aks show --name ${CLUSTER_NAME} --resource-group ${RESOURCE_GROUP} --query
"apiServerAccessProfile.authorizedIpRanges"
```

This command queries for the authorizedIpRanges property within the apiServerAccessProfile of your AKS cluster. The output will list the IP ranges that are authorized to access the AKS cluster's API server over the internet. If the list is empty, it means there are no restrictions, and any IP can access the AKS cluster's API server, assuming other network and security configurations allow it.

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 AKS 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):

Default Value:

By default, Endpoint Private Access is disabled.

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-network-security#ns-1-implement-security-for-internal-traffic

CIS Controls:

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

5.4.2 Ensure clusters are Private Cluster enabled and Public Access Disabled

Profile Applicability:

- Level 2

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 wirtual 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 virtual network.

Although Kubernetes API requires an authorized token to perform sensitive actions, a vulnerability could potentially expose the Kubernetes publically 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 virtual network to perform any attack on the Kubernetes API.

Audit:

Check for the following to be 'enabled: **false**'

```
export CLUSTER_NAME=<your cluster name>
export RESOURCE_GROUP=<your resource group name>

az aks show --name ${CLUSTER_NAME} --resource-group ${RESOURCE_GROUP} --query
"apiServerAccessProfile.enablePublicFqdn"
```

This command queries for the enablePublicFqdn property within the apiServerAccessProfile of your AKS cluster. The output will be true if endpointPublicAccess is enabled, allowing access to the AKS cluster API server from the internet. If it's false, endpointPublicAccess is disabled, meaning the API server is not accessible over the internet, which is a common configuration for private clusters.
Check for the following to be 'enabled: **true**'

```
az aks show --name ${CLUSTER_NAME} --resource-group ${RESOURCE_GROUP} --query
"apiServerAccessProfile.enablePrivateCluster"
```

This command queries the enablePrivateCluster property within the apiServerAccessProfile of your AKS cluster. If the output is true, it indicates that endpointPrivateAccess is enabled, and the AKS cluster API server is configured to be accessible only via a private endpoint. If the output is false, the cluster is not configured for private access only, and the API server might be accessible over the internet depending on other settings.

Remediation:

To use a private endpoint, create a new private endpoint in your virtual network then create a link between your virtual network and a new private DNS zone

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-network-security#ns-2-connect-private-networks-together
2. https://learn.microsoft.com/en-us/azure/aks/private-clusters

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 to be 'enabled: true'

```
export CLUSTER_NAME=<your cluster name>
export RESOURCE_GROUP=<your resource group name>

az aks show --name ${CLUSTER_NAME} --resource-group ${RESOURCE_GROUP} --query
"apiServerAccessProfile.enablePrivateCluster"
```

Check for the following is not null:

```
az aks show --name ${CLUSTER_NAME} --resource-group ${RESOURCE_GROUP} --query
"apiServerAccessProfile.authorizedIpRanges"
```

Remediation:

```
az aks create \
--resource-group <private-cluster-resource-group> \
--name <private-cluster-name> \
--load-balancer-sku standard \
--enable-private-cluster \
--network-plugin azure \
--vnet-subnet-id <subnet-id> \
--docker-bridge-address \
--dns-service-ip \
--service-cidr
```

Where `--enable-private-cluster` is a mandatory flag for a private cluster.

References:

1. https://learn.microsoft.com/en-us/azure/aks/private-clusters

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 Network Policy is Enabled and set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

When you run modern, microservices-based applications in Kubernetes, you often want to control which components can communicate with each other. The principle of least privilege should be applied to how traffic can flow between pods in an Azure Kubernetes Service (AKS) cluster. Let's say you likely want to block traffic directly to back-end applications. The Network Policy feature in Kubernetes lets you define rules for ingress and egress traffic between pods in a cluster.

Rationale:

All pods in an AKS cluster can send and receive traffic without limitations, by default. To improve security, you can define rules that control the flow of traffic. Back-end applications are often only exposed to required front-end services, for example. Or, database components are only accessible to the application tiers that connect to them.

Network Policy is a Kubernetes specification that defines access policies for communication between Pods. Using Network Policies, you define an ordered set of rules to send and receive traffic and apply them to a collection of pods that match one or more label selectors.

These network policy rules are defined as YAML manifests. Network policies can be included as part of a wider manifest that also creates a deployment or service.

Impact:

Network Policy requires the Network Policy add-on. This add-on is included automatically when a cluster with Network Policy is created, but for an existing cluster, needs to be added prior to enabling Network Policy.

Enabling/Disabling Network Policy causes a rolling update of all cluster nodes, similar to performing a cluster upgrade. This operation is long-running and will block other operations on the cluster (including delete) until it has run to completion.

If Network Policy is used, a cluster must have at least 2 nodes of type `n1-standard-1` or higher. The recommended minimum size cluster to run Network Policy enforcement is 3 `n1-standard-1` instances.

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 not null and set with appropriate group id:

```
export CLUSTER_NAME=<your cluster name>
az aks show --name ${CLUSTER_NAME} --resource-group ${RESOURCE_GROUP} --query
"networkProfile.networkPolicy"
```

Remediation:

Utilize Calico or other network policy engine to segment and isolate your traffic.

Default Value:

By default, Network Policy is disabled.

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-network-security#ns-2-connect-private-networks-together
2. https://docs.microsoft.com/en-us/azure/aks/use-network-policies

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.	●	●	●
v7	9.2 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 2

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:

Remediation:

References:

1. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-data-protection#dp-4-encrypt-sensitive-information-in-transit

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 Azure (AKS).

5.5.1 Manage Kubernetes RBAC users with Azure AD (Manual)

Profile Applicability:

- Level 2

Description:

Azure Kubernetes Service (AKS) can be configured to use Azure Active Directory (AD) for user authentication. In this configuration, you sign in to an AKS cluster using an Azure AD authentication token. You can also configure Kubernetes role-based access control (Kubernetes RBAC) to limit access to cluster resources based a user's identity or group membership.

Rationale:

Kubernetes RBAC and AKS help you secure your cluster access and provide only the minimum required permissions to developers and operators.

Audit:

Remediation:

References:

1. https://docs.microsoft.com/en-us/azure/aks/azure-ad-rbac
2. https://docs.microsoft.com/security/benchmark/azure/security-controls-v2-privileged-access#pa-7-follow-just-enough-administration-least-privilege-principle

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.		●	●

5.5.2 Use Azure RBAC for Kubernetes Authorization (Manual)

Profile Applicability:

- Level 2

Description:

The ability to manage RBAC for Kubernetes resources from Azure gives you the choice to manage RBAC for the cluster resources either using Azure or native Kubernetes mechanisms. When enabled, Azure AD principals will be validated exclusively by Azure RBAC while regular Kubernetes users and service accounts are exclusively validated by Kubernetes RBAC.

Azure role-based access control (RBAC) is an authorization system built on Azure Resource Manager that provides fine-grained access management of Azure resources.

With Azure RBAC, you create a role definition that outlines the permissions to be applied. You then assign a user or group this role definition via a role assignment for a particular scope. The scope can be an individual resource, a resource group, or across the subscription.

Rationale:

Today you can already leverage integrated authentication between Azure Active Directory (Azure AD) and AKS. When enabled, this integration allows customers to use Azure AD users, groups, or service principals as subjects in Kubernetes RBAC. This feature frees you from having to separately manage user identities and credentials for Kubernetes. However, you still have to set up and manage Azure RBAC and Kubernetes RBAC separately. Azure RBAC for Kubernetes Authorization is an approach that allows for the unified management and access control across Azure Resources, AKS, and Kubernetes resources.

Audit:

Remediation:

References:

1. https://docs.microsoft.com/en-us/azure/aks/manage-azure-rbac

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**Master (Control Plane) Components**		
2	**Master (Control Plane) Configuration**		
2.1	**Logging**		
2.1.1	Enable audit Logs (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 azure.json file has permissions set to 644 or more restrictive (Automated)	☐	☐
3.1.4	Ensure that the azure.json file ownership is set to root:root (Automated)	☐	☐
3.2	**Kubelet**		
3.2.1	Ensure that the --anonymous-auth argument is set to false (Automated)	☐	☐
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	☐	☐
3.2.3	Ensure that the --client-ca-file argument is set as appropriate (Automated)	☐	☐
3.2.4	Ensure that the --read-only-port is secured (Automated)	☐	☐
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
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 set to false (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 (Automated)	☐	☐
4.1.2	Minimize access to secrets (Automated)	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles (Automated)	☐	☐
4.1.4	Minimize access to create pods (Automated)	☐	☐
4.1.5	Ensure that default service accounts are not actively used (Automated)	☐	☐
4.1.6	Ensure that Service Account Tokens are only mounted where necessary (Automated)	☐	☐
4.2	**Pod Security Standards**		
4.2.1	Minimize the admission of privileged containers (Automated)	☐	☐
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace (Automated)	☐	☐
4.2.4	Minimize the admission of containers wishing to share the host network namespace (Automated)	☐	☐
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation (Automated)	☐	☐
4.3	**Azure Policy / OPA**		
4.4	**CNI Plugin**		
4.4.1	Ensure latest CNI version is used (Automated)	☐	☐
4.4.2	Ensure that all Namespaces have Network Policies defined (Automated)	☐	☐
4.5	**Secrets Management**		
4.5.1	Prefer using secrets as files over secrets as environment variables (Automated)	☐	☐
4.5.2	Consider external secret storage (Manual)	☐	☐
4.6	**General Policies**		
4.6.1	Create administrative boundaries between resources using namespaces (Manual)	☐	☐
4.6.2	Apply Security Context to Your Pods and Containers (Manual)	☐	☐
4.6.3	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 Azure Defender image scanning or a third party provider (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.1.2	Minimize user access to Azure Container Registry (ACR) (Manual)	☐	☐
5.1.3	Minimize cluster access to read-only for Azure Container Registry (ACR) (Manual)	☐	☐
5.1.4	Minimize Container Registries to only those approved (Manual)	☐	☐
5.2	**Access and identity options for Azure Kubernetes Service (AKS)**		
5.2.1	Prefer using dedicated AKS Service Accounts (Manual)	☐	☐
5.3	**Key Management Service (KMS)**		
5.3.1	Ensure Kubernetes Secrets are encrypted (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)	☐	☐
5.4.4	Ensure Network Policy 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 Azure AD (Manual)	☐	☐
5.5.2	Use Azure RBAC for Kubernetes Authorization (Manual)	☐	☐

Appendix: CIS Controls v7 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.5	Ensure that default service accounts are not actively used	☐	☐
4.6.3	The default namespace should not be used	☐	☐
5.1.2	Minimize user access to Azure Container Registry (ACR)	☐	☐
5.2.1	Prefer using dedicated AKS Service Accounts	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.4	Ensure Network Policy is Enabled and set as appropriate	☐	☐
5.5.2	Use Azure RBAC for Kubernetes Authorization	☐	☐

Appendix: CIS Controls v7 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
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 azure.json file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the azure.json file ownership is set to root:root	☐	☐
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
3.2.4	Ensure that the --read-only-port is secured	☐	☐
3.2.8	Ensure that the --rotate-certificates argument is not set to false	☐	☐
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	☐	☐
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	☐	☐
4.2.1	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	☐	☐

Recommendation		Set Correctly	
		Yes	No
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	☐	☐
4.4.1	Ensure latest CNI version is used	☐	☐
4.4.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.5.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.6.3	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning using Azure Defender image scanning or a third party provider	☐	☐
5.1.2	Minimize user access to Azure Container Registry (ACR)	☐	☐
5.1.3	Minimize cluster access to read-only for Azure Container Registry (ACR)	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.2.1	Prefer using dedicated AKS Service Accounts	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.4	Ensure Network Policy 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 Azure AD	☐	☐
5.5.2	Use Azure RBAC for Kubernetes Authorization	☐	☐

Appendix: CIS Controls v7 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
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 azure.json file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the azure.json file ownership is set to root:root	☐	☐
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	☐	☐
3.2.4	Ensure that the --read-only-port is secured	☐	☐
3.2.8	Ensure that the --rotate-certificates argument is not set to false	☐	☐
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	☐	☐
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	☐	☐
4.2.1	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	☐	☐

Recommendation		Set Correctly	
		Yes	No
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	☐	☐
4.4.1	Ensure latest CNI version is used	☐	☐
4.4.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.5.1	Prefer using secrets as files over secrets as environment variables	☐	☐
4.5.2	Consider external secret storage	☐	☐
4.6.3	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning using Azure Defender image scanning or a third party provider	☐	☐
5.1.2	Minimize user access to Azure Container Registry (ACR)	☐	☐
5.1.3	Minimize cluster access to read-only for Azure Container Registry (ACR)	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.2.1	Prefer using dedicated AKS Service Accounts	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted	☐	☐
5.4.1	Restrict Access to the Control Plane Endpoint	☐	☐
5.4.4	Ensure Network Policy 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 Azure AD	☐	☐
5.5.2	Use Azure RBAC for Kubernetes Authorization	☐	☐

Appendix: CIS Controls v7 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
	No unmapped recommendations to CIS Controls v7.0	☐	☐

Appendix: CIS Controls v8 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Enable audit Logs	☐	☐
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 azure.json file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the azure.json file ownership is set to root:root	☐	☐
3.2.4	Ensure that the --read-only-port is secured	☐	☐
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	☐	☐
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	☐	☐
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	☐	☐
4.2.1	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	☐	☐

Recommendation		Set Correctly	
		Yes	No
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	☐	☐
4.6.1	Create administrative boundaries between resources using namespaces	☐	☐
4.6.2	Apply Security Context to Your Pods and Containers	☐	☐
4.6.3	The default namespace should not be used	☐	☐
5.1.2	Minimize user access to Azure Container Registry (ACR)	☐	☐
5.1.3	Minimize cluster access to read-only for Azure Container Registry (ACR)	☐	☐
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 Network Policy is Enabled and set as appropriate	☐	☐

Appendix: CIS Controls v8 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Enable audit Logs	☐	☐
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 azure.json file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the azure.json file ownership is set to root:root	☐	☐
3.2.4	Ensure that the --read-only-port is secured	☐	☐
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 set to false	☐	☐
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	☐	☐
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	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐

Recommendation		Set Correctly	
		Yes	No
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	☐	☐
4.4.1	Ensure latest CNI version is used	☐	☐
4.4.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.6.1	Create administrative boundaries between resources using namespaces	☐	☐
4.6.2	Apply Security Context to Your Pods and Containers	☐	☐
4.6.3	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning using Azure Defender image scanning or a third party provider	☐	☐
5.1.2	Minimize user access to Azure Container Registry (ACR)	☐	☐
5.1.3	Minimize cluster access to read-only for Azure Container Registry (ACR)	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted	☐	☐
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 Network Policy 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	☐	☐
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 azure.json file has permissions set to 644 or more restrictive	☐	☐
3.1.4	Ensure that the azure.json file ownership is set to root:root	☐	☐
3.2.4	Ensure that the --read-only-port is secured	☐	☐
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 set to false	☐	☐
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	☐	☐
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	☐	☐
4.2.1	Minimize the admission of privileged containers	☐	☐

Recommendation		Set Correctly	
		Yes	No
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	☐	☐
4.4.1	Ensure latest CNI version is used	☐	☐
4.4.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.6.1	Create administrative boundaries between resources using namespaces	☐	☐
4.6.2	Apply Security Context to Your Pods and Containers	☐	☐
4.6.3	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning using Azure Defender image scanning or a third party provider	☐	☐
5.1.2	Minimize user access to Azure Container Registry (ACR)	☐	☐
5.1.3	Minimize cluster access to read-only for Azure Container Registry (ACR)	☐	☐
5.1.4	Minimize Container Registries to only those approved	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted	☐	☐
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 Network Policy 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 Azure AD	☐	☐
5.5.2	Use Azure RBAC for Kubernetes Authorization	☐	☐

Appendix: CIS Controls v8 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
	No unmapped recommendations to CIS Controls v8.0	☐	☐

Appendix: Change History

Date	Version	Changes for this version
March 19, 2024	1.5.0	Support for Kubernetes Clusters created on 1.29, 1.28, 1.27
March 10, 2024	1.5.0	Automated Section 4.2 – Pod Security Standards recommendations to use CIS-CAT
March 1, 2024	1.5.0	Automated Section 4.1 - RBAC and Service Accounts recommendations to use CIS-CAT
February 29, 2024	1.5.0	Automated Section 5.4 – Cluster Networking recommendations to use CIS-CAT
February 21, 2024	1.5.0	Automated Section 3.1 – Worker node Configuration File recommendations to use CIS-CAT
Sept 29, 2023	1.4.0	Support for Kubernetes Clusters created on v1.27, 1.26 or 1.25
Sept 27, 2023	1.4.0	Ticket # 19278 Addressing – eventRecordQPS flag and title
Sept 27, 2023	1.4.0	Ticket # 19279 Validated recommendation 3.2.10 and the methods that can be utilized to audit the setting.

Date	Version	Changes for this version
Sept 15, 2023	1.4.0	Ticket #19350 Validated recommendation for -streaming-connection-idle-timeout
Apr 9, 2023	1.3.0	Ticket # 17664 The current state of check AKS 3.2.9 is at best undefined: "set to 0 or a level which ensures appropriate event capture".
Apr 9, 2023	1.3.0	Ticket #16491 Edited recommendation 5.2.10
Apr 3, 2023	1.3.0	Ticket # 16624 Updated Recommendation 5.3.2 moved flags in audit process to end of the command line
Apr 3, 2023	1.3.0	Ticket # 16625Updated Recommendation 5.7.1 default value statement.
Apr 3, 2023	1.3.0	Ticket #18522 Set Pod Security Policy Recommendations to Manual in preparation for PSP removal in v1.25 and beyond.
Mar 22, 2023	1.3.0	Ticket #17029 Recommendation 4.2.1 Minimize the admission of privileged containers is deprricated.

Date	Version	Changes for this version
Mar 13, 2023	1.3.0	Ticket #16686 Updated recommendation 4.2.4 to resolve conflict with 4.2.8
Mar 13, 2023	1.3.0	Ticket # 15917 edited –event-qps option with eventRecordQPS