# UNCLASSIFIED

# Kubernetes Security Technical Implementation Guide

## Version: 1

## Release: 4

## 27 Jan 2022

**XSL Release 11/7/2019 Sort by: STIGID**
**Description:** This Security Technical Implementation Guide is published as a tool to improve the security of Department of Defense (DoD) information systems. The requirements are derived from the National Institute of Standards and Technology (NIST) 800-53 and related documents. Comments or proposed revisions to this document should be sent via email to the following address: disa.stig_spt@mail.mil.

---

**Group ID (Vulid):** V-242376
**Group Title:** SRG-APP-000014-CTR-000035
**Rule ID:** SV-242376r712484_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000150

**Rule Title:** The Kubernetes Controller Manager must use TLS 1.2, at a minimum, to protect the confidentiality of sensitive data during electronic dissemination.

**Vulnerability Discussion:** The Kubernetes Controller Manager will prohibit the use of SSL and unauthorized versions of TLS protocols to properly secure communication.

The use of unsupported protocol exposes vulnerabilities to the Kubernetes by rogue traffic interceptions, man-in-the-middle attacks, and impersonation of users or services from the container platform runtime, registry, and key store. To enable the minimum version of TLS to be used by the Kubernetes Controller Manager, the setting "tls-min-version" must be set.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

grep -i tls-min-version *

If the setting "tls-min-version" is not configured in the Kubernetes Controller Manager manifest file or it is set to "VersionTLS10" or "VersionTLS11", this is a finding.

**Fix Text:** Edit the Kubernetes Controller Manager manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--tls-min-version" to "VersionTLS12" or higher.

**CCI:** CCI-000068

---

**Group ID (Vulid):** V-242377
**Group Title:** SRG-APP-000014-CTR-000035
**Rule ID:** SV-242377r712487_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000160
**Rule Title:** The Kubernetes Scheduler must use TLS 1.2, at a minimum, to protect the confidentiality of sensitive data during electronic dissemination.

**Vulnerability Discussion:** The Kubernetes Scheduler will prohibit the use of SSL and unauthorized versions of TLS protocols to properly secure communication.

The use of unsupported protocol exposes vulnerabilities to the Kubernetes by rogue traffic interceptions, man-in-the-middle attacks, and impersonation of users or services from the container platform runtime, registry, and keystore. To enable the minimum version of TLS to be used by the Kubernetes API Server, the setting "tls-min-version" must be set.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

grep -i tls-min-version *

If the setting "tls-min-version" is not configured in the Kubernetes Scheduler manifest file or it is set to "VersionTLS10" or "VersionTLS11", this is a finding.

**Fix Text:** Edit the Kubernetes Scheduler manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--tls-min-version" to "VersionTLS12" or higher.

**CCI:** CCI-000068

---

**Group ID (Vulid):** V-242378
**Group Title:** SRG-APP-000014-CTR-000040
**Rule ID:** SV-242378r712490_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000170
**Rule Title:** The Kubernetes API Server must use TLS 1.2, at a minimum, to protect the confidentiality of sensitive data during electronic dissemination.

**Vulnerability Discussion:** The Kubernetes API Server will prohibit the use of SSL and unauthorized versions of TLS protocols to properly secure communication.

The use of unsupported protocol exposes vulnerabilities to the Kubernetes by rogue traffic interceptions, man-in-the-middle attacks, and impersonation of users or services from the container platform runtime, registry, and keystore. To enable the minimum version of TLS to be used by the Kubernetes API Server, the setting "tls-min-version" must be set.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i tls-min-version *
```

If the setting "tls-min-version" is not configured in the Kubernetes API Server manifest file or it is set to "VersionTLS10" or "VersionTLS11", this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--tls-min-version" to "VersionTLS12" or higher.

**CCI:** CCI-000068

---

**Group ID (Vulid):** V-242379
**Group Title:** SRG-APP-000014-CTR-000035
**Rule ID:** SV-242379r754799_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000180
**Rule Title:** The Kubernetes etcd must use TLS to protect the confidentiality of sensitive data during electronic dissemination.

**Vulnerability Discussion:** Kubernetes etcd will prohibit the use of SSL and unauthorized versions of TLS protocols to properly secure communication.

The use of unsupported protocol exposes vulnerabilities to the Kubernetes by rogue traffic interceptions, man-in-the-middle attacks, and impersonation of users or services from the container platform runtime, registry, and keystore. To enable the minimum version of TLS to be used by the Kubernetes API Server, the setting "auto-tls" must be set.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i auto-tls *
```

If the setting "auto-tls" is not configured in the Kubernetes etcd manifest file or it is set to true, this is a finding.

**Fix Text:** Edit the Kubernetes etcd manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "-auto-tls" to "false".

**CCI:** CCI-000068

---

**Group ID (Vulid):** V-242380
**Group Title:** SRG-APP-000014-CTR-000035
**Rule ID:** SV-242380r754800_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000190
**Rule Title:** The Kubernetes etcd must use TLS to protect the confidentiality of sensitive data during electronic dissemination.

**Vulnerability Discussion:** The Kubernetes API Server will prohibit the use of SSL and unauthorized versions of TLS protocols to properly secure communication.

The use of unsupported protocol exposes vulnerabilities to the Kubernetes by rogue traffic interceptions, man-in-the-middle attacks, and impersonation of users or services from the container platform runtime, registry, and keystore. To enable the minimum version of TLS to be used by the Kubernetes API Server, the setting "peer-auto-tls" must be set.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -I peer-auto-tls *
```

If the setting "peer-auto-tls" is not configured in the Kubernetes etcd manifest file or it is set to "true", this is a finding.

**Fix Text:** Edit the Kubernetes etcd manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "peer-auto-tls" to "false".

**CCI:** CCI-000068

---

**Group ID (Vulid):** V-242381
**Group Title:** SRG-APP-000023-CTR-000055
**Rule ID:** SV-242381r799981_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-000220
**Rule Title:** The Kubernetes Controller Manager must create unique service accounts for each work payload.

**Vulnerability Discussion:** The Kubernetes Controller Manager is a background process that embeds core control loops regulating cluster system state through the API Server. Every process executed in a pod has an associated service account. By default, service accounts use the same credentials for authentication. Implementing the default settings poses a High risk to the Kubernetes Controller Manager. Setting the use-service-account-credential value lowers the attack surface by generating unique service accounts settings for each controller instance.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i use-service-account-credentials *

If the setting use-service-account-credentials is not configured in the Kubernetes Controller Manager manifest file or it is set to "false", this is a finding.

**Fix Text:** Edit the Kubernetes Controller Manager manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "use-service-account-credentials" to "true".

**CCI:** CCI-000015

---

**Group ID (Vulid):** V-242382
**Group Title:** SRG-APP-000033-CTR-000090
**Rule ID:** SV-242382r712502_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000270
**Rule Title:** The Kubernetes API Server must enable Node,RBAC as the authorization mode.

**Vulnerability Discussion:** To mitigate the risk of unauthorized access to sensitive information by entities that have been issued certificates by DoD-approved PKIs, all DoD systems (e.g., networks, web servers, and web portals) must be properly configured to incorporate access control methods that do not rely solely on the possession of a certificate for access. Successful authentication must not automatically give an entity access to an asset or security boundary. Authorization procedures and controls must be implemented to ensure each authenticated entity also has a validated and current authorization. Authorization is the process of determining whether an entity, once authenticated, is permitted to access a specific asset.

Node,RBAC is the method within Kubernetes to control access of users and applications. Kubernetes uses roles to grant authorization API requests made by kubelets.

Satisfies: SRG-APP-000033-CTR-000090, SRG-APP-000033-CTR-000095

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

"grep -i authorization-mode *"

If the setting "authorization-mode" is not configured in the Kubernetes API Server manifest file or is not set to "Node,RBAC", this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--authorization-mode" to "Node,RBAC".

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242383
**Group Title:** SRG-APP-000038-CTR-000105
**Rule ID:** SV-242383r712505_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-000290
**Rule Title:** User-managed resources must be created in dedicated namespaces.

**Vulnerability Discussion:** Creating namespaces for user-managed resources is important when implementing

Role-Based Access Controls (RBAC). RBAC allows for the authorization of users and helps support proper API server permissions separation and network micro segmentation. If user-managed resources are placed within the default namespaces, it becomes impossible to implement policies for RBAC permission, service account usage, network policies, and more.

**Check Content:**
To view the available namespaces, run the command:

```
kubectl get namespaces
```

The default namespaces to be validated are default, kube-public and kube-node-lease if it is created.

For the default namespace, execute the commands:

```
kubectl config set-context --current --namespace=default
kubectl get all
```

For the kube-public namespace, execute the commands:

```
kubectl config set-context --current --namespace=kube-public
kubectl get all
```

For the kube-node-lease namespace, execute the commands:

```
kubectl config set-context --current --namespace=kube-node-lease
kubectl get all
```

The only valid return values are the kubernetes service (i.e., service/kubernetes) and nothing at all.

If a return value is returned from the "kubectl get all" command and it is not the kubernetes service (i.e., service/kubernetes), this is a finding.

**Fix Text:** Move any user-managed resources from the default, kube-public and kube-node-lease namespaces, to user namespaces.

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242384
**Group Title:** SRG-APP-000033-CTR-000090
**Rule ID:** SV-242384r712508_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000300
**Rule Title:** The Kubernetes Scheduler must have secure binding.

**Vulnerability Discussion:** Limiting the number of attack vectors and implementing authentication and encryption on the endpoints available to external sources is paramount when securing the overall Kubernetes cluster. The Scheduler API service exposes port 10251/TCP by default for health and metrics information use. This port does not encrypt or authenticate connections. If this port is exposed externally, an attacker can use this port to attack the entire Kubernetes cluster. By setting the bind address to localhost (i.e., 127.0.0.1), only those internal services that require health and metrics information can access the Scheduler API.

**Check Content:**

Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i bind-address *

If the setting "bind-address" is not set to "127.0.0.1" or is not found in the Kubernetes Scheduler manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes Scheduler manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the argument "--bind-address" to "127.0.0.1".

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242385
**Group Title:** SRG-APP-000033-CTR-000090
**Rule ID:** SV-242385r712511_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000310
**Rule Title:** The Kubernetes Controller Manager must have secure binding.

**Vulnerability Discussion:** Limiting the number of attack vectors and implementing authentication and encryption on the endpoints available to external sources is paramount when securing the overall Kubernetes cluster. The Controller Manager API service exposes port 10252/TCP by default for health and metrics information use. This port does not encrypt or authenticate connections. If this port is exposed externally, an attacker can use this port to attack the entire Kubernetes cluster. By setting the bind address to only localhost (i.e., 127.0.0.1), only those internal services that require health and metrics information can access the Control Manager API.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i bind-address *

If the setting bind-address is not set to "127.0.0.1" or is not found in the Kubernetes Controller Manager manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes Controller Manager manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the argument "--bind-address" to "127.0.0.1".

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242386
**Group Title:** SRG-APP-000033-CTR-000095
**Rule ID:** SV-242386r808574_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-000320
**Rule Title:** The Kubernetes API server must have the insecure port flag disabled.

**Vulnerability Discussion:** By default, the API server will listen on two ports. One port is the secure port and the other port is called the "localhost port". This port is also called the "insecure port", port 8080. Any requests to this port bypass authentication and authorization checks. If this port is left open, anyone who gains access to the host on which the master is running can bypass all authorization and authentication mechanisms put in place, and have full control over the entire cluster.

Close the insecure port by setting the API server's --insecure-port flag to "0", ensuring that the --insecure-bind-address is not set.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i insecure-port *

If the setting insecure-port is not set to "0" or is not configured in the Kubernetes API server manifest file, this is a finding.

NOTE: --insecure-port flag has been deprecated and can only be set to 0, **This flag will be removed in v1.24.*

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the argument --insecure-port to "0".

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242387
**Group Title:** SRG-APP-000033-CTR-000095
**Rule ID:** SV-242387r717013_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-000330
**Rule Title:** The Kubernetes Kubelet must have the read-only port flag disabled.

**Vulnerability Discussion:** Kubelet serves a small REST API with read access to port 10255. The read-only port for Kubernetes provides no authentication or authorization security control. Providing unrestricted access on port 10255 exposes Kubernetes pods and containers to malicious attacks or compromise. Port 10255 is deprecated and should be disabled.

Close the read-only-port by setting the API server's read-only port flag to "0".

**Check Content:**
Run the following command on each Worker Node:
ps -ef | grep kubelet

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

If the --read-only-port argument exists and is not set to "0", this is a finding.

If the --read-only-port argument does not exist, check the Master Node Kubelet config file.

On the Kubernetes Master Node, run the command:
ps -ef | grep kubelet
(path identified by: --config)

Verify there is a readOnlyPort entry in the config file and it is set to "0".

If the --read-only-port argument exists and is not set to "0" this is a finding.

If "--read-only-port=0" argument does not exist on the worker node and the master node, this is a finding.

**Fix Text:** Edit the Kubernetes Kubelet file in the --config directory on the Kubernetes Master Node. Set the argument --read-only-port to 0.

Reset Kubelet service using the following command:
service kubelet restart

If using worker node arguments, edit the kubelet service file /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf on each Worker Node: set the parameter in KUBELET_SYSTEM_PODS_ARGS variable to "--read-only-port=0".

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242388
**Group Title:** SRG-APP-000033-CTR-000095
**Rule ID:** SV-242388r712520_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-000340
**Rule Title:** The Kubernetes API server must have the insecure bind address not set.

**Vulnerability Discussion:** By default, the API server will listen on two ports and addresses. One address is the secure address and the other address is called the "insecure bind" address and is set by default to localhost. Any requests to this address bypass authentication and authorization checks. If this insecure bind address is set to localhost, anyone who gains access to the host on which the master is running can bypass all authorization and authentication mechanisms put in place and have full control over the entire cluster.

Close or set the insecure bind address by setting the API server's --insecure-bind-address flag to an IP or leave it unset and ensure that the --insecure-bind-port is not set.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i insecure-bind-address *

If the setting insecure-bind-address is found and set to "localhost" in the Kubernetes API manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Remove the value for the --insecure-bind-address setting.

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242389
**Group Title:** SRG-APP-000033-CTR-000100
**Rule ID:** SV-242389r712523_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000350
**Rule Title:** The Kubernetes API server must have the secure port set.

**Vulnerability Discussion:** By default, the API server will listen on what is rightfully called the secure port, port

6443. Any requests to this port will perform authentication and authorization checks. If this port is disabled, anyone who gains access to the host on which the master is running has full control of the entire cluster over encrypted traffic.

Open the secure port by setting the API server's --secure-port flag to a value other than "0".

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i secure-port *

If the setting secure-port is set to "0" or is not configured in the Kubernetes API manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the argument --secure-port to a value greater than "0".

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242390
**Group Title:** SRG-APP-000033-CTR-000100
**Rule ID:** SV-242390r712526_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-000360
**Rule Title:** The Kubernetes API server must have anonymous authentication disabled.

**Vulnerability Discussion:** The Kubernetes API Server controls Kubernetes via an API interface. A user who has access to the API essentially has root access to the entire Kubernetes cluster. To control access, users must be authenticated and authorized. By allowing anonymous connections, the controls put in place to secure the API can be bypassed.

Setting anonymous authentication to "false" also disables unauthenticated requests from kubelets.

While there are instances where anonymous connections may be needed (e.g., health checks) and Role-Based Access Controls (RBAC) are in place to limit the anonymous access, this access should be disabled, and only enabled when necessary.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i anonymous-auth *

If the setting anonymous-auth is set to "true" in the Kubernetes API Server manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the argument --anonymous-auth to "false".

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242391
**Group Title:** SRG-APP-000033-CTR-000090
**Rule ID:** SV-242391r712529_rule
**Severity: CAT I**

**Rule Version (STIG-ID):** CNTR-K8-000370
**Rule Title:** The Kubernetes Kubelet must have anonymous authentication disabled.

**Vulnerability Discussion:** A user who has access to the Kubelet essentially has root access to the nodes contained within the Kubernetes Control Plane. To control access, users must be authenticated and authorized. By allowing anonymous connections, the controls put in place to secure the Kubelet can be bypassed.

Setting anonymous authentication to "false" also disables unauthenticated requests from kubelets.

While there are instances where anonymous connections may be needed (e.g., health checks) and Role-Based Access Controls (RBAC) are in place to limit the anonymous access, this access must be disabled and only enabled when necessary.

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node. Run the command:

grep -i anonymous-auth kubelet

If the setting "anonymous-auth" is set to "true" or the parameter not set in the Kubernetes Kubelet configuration file, this is a finding.

**Fix Text:** Edit the Kubernetes Kubelet file in the/etc/sysconfig/ directory on the Kubernetes Master Node.

Set the argument "--anonymous-auth" to "false".

Restart kubelet service using command:
service kubelet restart

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242392
**Group Title:** SRG-APP-000033-CTR-000095
**Rule ID:** SV-242392r712532_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-000380
**Rule Title:** The Kubernetes kubelet must enable explicit authorization.

**Vulnerability Discussion:** Kubelet is the primary agent on each node. The API server communicates with each kubelet to perform tasks such as starting/stopping pods. By default, kubelets allow all authenticated requests, even anonymous ones, without requiring any authorization checks from the API server. This default behavior bypasses any authorization controls put in place to limit what users may perform within the Kubernetes cluster. To change this behavior, the default setting of AlwaysAllow for the authorization mode must be set to "Webhook".

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node. Run the command:

grep -i authorization-mode kubelet

On each Worker node, change to the /etc/sysconfig/ directory. Run the command:

grep -i authorization-mode kubelet

If authorization-mode is missing or is set to "AllowAlways" on the Master node or any of the Worker nodes, this is a finding.

**Fix Text:** Edit the Kubernetes Kubelet file in the/etc/sysconfig/ directory on the Kubernetes Master and Worker nodes.

Set the argument --authorization-mode to "Webhook".

Restart each kubelet service after the change is made using the command:
service kubelet restart

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242393
**Group Title:** SRG-APP-000033-CTR-000095
**Rule ID:** SV-242393r717015_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000400
**Rule Title:** Kubernetes Worker Nodes must not have sshd service running.

**Vulnerability Discussion:** Worker Nodes are maintained and monitored by the Master Node. Direct access and manipulation of the nodes should not take place by administrators. Worker nodes should be treated as immutable and updated via replacement rather than in-place upgrades.

**Check Content:**
Log in to each worker node. Verify that the sshd service is not running. To validate that the service is not running, run the command:

systemctl status sshd

If the service sshd is active (running), this is a finding.

Note: If console access is not available, SSH access can be attempted. If the worker nodes cannot be reached, this requirement is "not a finding".

**Fix Text:** To stop the sshd service, run the command:

systemctl stop sshd

Note: If access to the worker node is through an SSH session, it is important to realize there are two requirements for disabling and stopping the sshd service and they should be done during the same SSH session. Disabling the service must be performed first and then the service stopped to guarantee both settings can be made if the session is interrupted.

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242394
**Group Title:** SRG-APP-000033-CTR-000095
**Rule ID:** SV-242394r717017_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000410

**Rule Title:** Kubernetes Worker Nodes must not have the sshd service enabled.

**Vulnerability Discussion:** Worker Nodes are maintained and monitored by the Master Node. Direct access and manipulation of the nodes must not take place by administrators. Worker nodes must be treated as immutable and updated via replacement rather than in-place upgrades.

**Check Content:**
Log in to each worker node. Verify that the sshd service is not enabled. To validate the service is not enabled, run the command:

```
systemctl is-enabled sshd.service
```

If the service sshd is enabled, this is a finding.

Note: If console access is not available, SSH access can be attempted. If the worker nodes cannot be reached, this requirement is "not a finding".

**Fix Text:** To disable the sshd service, run the command:

```
chkconfig sshd off
```

Note: If access to the worker node is through an SSH session, it is important to realize there are two requirements for disabling and stopping the sshd service that must be done during the same SSH session. Disabling the service must be performed first and then the service stopped to guarantee both settings can be made if the session is interrupted.

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242395
**Group Title:** SRG-APP-000033-CTR-000095
**Rule ID:** SV-242395r712541_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000420
**Rule Title:** Kubernetes dashboard must not be enabled.

**Vulnerability Discussion:** While the Kubernetes dashboard is not inherently insecure on its own, it is often coupled with a misconfiguration of Role-Based Access control (RBAC) permissions that can unintentionally over-grant access. It is not commonly protected with "NetworkPolicies", preventing all pods from being able to reach it. In increasingly rare circumstances, the Kubernetes dashboard is exposed publicly to the internet.

**Check Content:**
From the master node, run the command:

```
kubectl get pods --all-namespaces -l k8s-app=kubernetes-dashboard
```

If any resources are returned, this is a finding.

**Fix Text:** Delete the Kubernetes dashboard deployment with the following command:

```
kubectl delete deployment kubernetes-dashboard --namespace=kube-system
```

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242396
**Group Title:** SRG-APP-000033-CTR-000090
**Rule ID:** SV-242396r712544_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000430
**Rule Title:** Kubernetes Kubectl cp command must give expected access and results.

**Vulnerability Discussion:** One of the tools heavily used to interact with containers in the Kubernetes cluster is kubectl. The command is the tool System Administrators used to create, modify, and delete resources. One of the capabilities of the tool is to copy files to and from running containers (i.e., kubectl cp). The command uses the "tar" command of the container to copy files from the container to the host executing the "kubectl cp" command. If the "tar" command on the container has been replaced by a malicious user, the command can copy files anywhere on the host machine. This flaw has been fixed in later versions of the tool. It is recommended to use kubectl versions newer than 1.12.9.

**Check Content:**
From the Master and each Worker node, check the version of kubectl by executing the command:

```
kubectl version --client
```

If the Master or any Work nodes are not using kubectl version 1.12.9 or newer, this is a finding.

**Fix Text:** Upgrade the Master and Worker nodes to the latest version of kubectl.

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242397
**Group Title:** SRG-APP-000033-CTR-000090
**Rule ID:** SV-242397r712547_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-000440
**Rule Title:** The Kubernetes kubelet static PodPath must not enable static pods.

**Vulnerability Discussion:** Allowing kubelet to set a staticPodPath gives containers with root access permissions to traverse the hosting filesystem. The danger comes when the container can create a manifest file within the /etc/kubernetes/manifests directory. When a manifest is created within this directory, containers are entirely governed by the Kubelet not the API Server. The container is not susceptible to admission control at all. Any containers or pods that are instantiated in this manner are called "static pods" and are meant to be used for pods such as the API server, scheduler, controller, etc., not workload pods that need to be governed by the API Server.

**Check Content:**
On the Master and Worker nodes, change to the /etc/sysconfig/ directory and run the command:

```
grep -i staticPodPath kubelet
```

If any of the nodes return a value for staticPodPath, this is a finding.

**Fix Text:** Edit the kubelet file on each node under the /etc/sysconfig directory to remove the staticPodPath setting and restart the kubelet service by executing the command:

```
service kubelet restart
```

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242398
**Group Title:** SRG-APP-000033-CTR-000100
**Rule ID:** SV-242398r717019_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000450
**Rule Title:** Kubernetes DynamicAuditing must not be enabled.

**Vulnerability Discussion:** Protecting the audit data from change or deletion is important when an attack occurs. One way an attacker can cover their tracks is to change or delete audit records. This will either make the attack unnoticeable or make it more difficult to investigate how the attack took place and what changes were made. The audit data can be protected through audit log file protections and user authorization.

One way for an attacker to thwart these measures is to send the audit logs to another source and filter the audited results before sending them on to the original target. This can be done in Kubernetes through the configuration of dynamic audit webhooks through the DynamicAuditing flag.

**Check Content:**
On the Master node, change to the manifests' directory at /etc/kubernetes/manifests and run the command:

grep -i feature-gates *

Review the feature-gates setting, if one is returned.

If the feature-gates setting is available and contains the DynamicAuditing flag set to "true", this is a finding.

Change to the directory /etc/sysconfig on the Master and each Worker Node and execute the command:

grep -i feature-gates kubelet

Review every feature-gates setting that is returned.

If any feature-gates setting is available and contains the "DynamicAuditing" flag set to "true", this is a finding.

**Fix Text:** Edit any manifest files or kubelet config files that contain the feature-gates setting with DynamicAuditing set to "true". Set the flag to "false" or remove the "DynamicAuditing" setting completely. Restart the kubelet service if the kubelet config file if the kubelet config file is changed.

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242399
**Group Title:** SRG-APP-000033-CTR-000095
**Rule ID:** SV-242399r717021_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000460
**Rule Title:** Kubernetes DynamicKubeletConfig must not be enabled.

**Vulnerability Discussion:** Kubernetes allows a user to configure kubelets with dynamic configurations. When dynamic configuration is used, the kubelet will watch for changes to the configuration file. When changes are made, the kubelet will automatically restart. Allowing this capability bypasses access restrictions and authorizations. Using this capability, an attacker can lower the security posture of the kubelet, which includes allowing the ability to run arbitrary commands in any container running on that node.

**Check Content:**
On the Master node, change to the manifests' directory at /etc/kubernetes/manifests and run the command:

grep -i feature-gates *

Review the feature-gates setting if one is returned.

If the feature-gates setting does not exist or feature-gates does not contain the DynamicKubeletConfig flag or the "DynamicKubletConfig" flag is set to "true", this is a finding.

Change to the directory /etc/sysconfig on the Master and each Worker node and execute the command:

grep -i feature-gates kubelet

Review every feature-gates setting if one is returned.

If the feature-gates setting does not exist or feature-gates does not contain the DynamicKubeletConfig flag or the DynamicKubletConfig flag is set to "true", this is a finding.

**Fix Text:** Edit any manifest file or kubelet config file that does not contain a feature-gates setting or has DynamicKubeletConfig set to "true".

An omission of DynamicKubeletConfig within the feature-gates defaults to true. Set DynamicKubeletConfig to "false". Restart the kubelet service if the kubelet config file is changed.

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242400
**Group Title:** SRG-APP-000033-CTR-000090
**Rule ID:** SV-242400r712556_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000470
**Rule Title:** The Kubernetes API server must have Alpha APIs disabled.

**Vulnerability Discussion:** Kubernetes allows alpha API calls within the API server. The alpha features are disabled by default since they are not ready for production and likely to change without notice. These features may also contain security issues that are rectified as the feature matures. To keep the Kubernetes cluster secure and stable, these alpha features must not be used.

**Check Content:**
On the Master node, change to the manifests' directory at /etc/kubernetes/manifests and run the command:

grep -i feature-gates *

Review the feature-gates setting, if one is returned.

If the feature-gates setting is available and contains the AllAlpha flag set to "true", this is a finding.

**Fix Text:** Edit any manifest files that contain the feature-gates setting with AllAlpha set to "true". Set the flag to

"false" or remove the AllAlpha setting completely.
(AllAlpha- default=false)

**CCI:** CCI-000213

---

**Group ID (Vulid):** V-242401
**Group Title:** SRG-APP-000092-CTR-000165
**Rule ID:** SV-242401r712559_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000600
**Rule Title:** The Kubernetes API Server must have an audit policy set.

**Vulnerability Discussion:** When Kubernetes is started, components and user services are started. For auditing startup events, and events for components and services, it is important that auditing begin on startup. Within Kubernetes, audit data for all components is generated by the API server. To enable auditing to begin, an audit policy must be defined for the events and the information to be stored with each event. It is also necessary to give a secure location where the audit logs are to be stored. If an audit log path is not specified, all audit data is sent to studio.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i audit-policy-file *
```

If the audit-policy-file is not set, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest and set "--audit-policy-file" to the audit policy file.

Note: If the API server is running as a Pod, then the manifest will also need to be updated to mount the host system filesystem where the audit policy file resides.

**CCI:** CCI-001464

---

**Group ID (Vulid):** V-242402
**Group Title:** SRG-APP-000092-CTR-000165
**Rule ID:** SV-242402r712562_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000610
**Rule Title:** The Kubernetes API Server must have an audit log path set.

**Vulnerability Discussion:** When Kubernetes is started, components and user services are started for auditing startup events, and events for components and services, it is important that auditing begin on startup. Within Kubernetes, audit data for all components is generated by the API server. To enable auditing to begin, an audit policy must be defined for the events and the information to be stored with each event. It is also necessary to give a secure location where the audit logs are to be stored. If an audit log path is not specified, all audit data is sent to studio.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i audit-log-path *
```

If the audit-log-path is not set, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest and set "--audit-log-path" to a secure location for the audit logs to be written.

Note: If the API server is running as a Pod, then the manifest will also need to be updated to mount the host system filesystem where the audit log file is to be written.

**CCI:** CCI-001464

---

**Group ID (Vulid):** V-242403
**Group Title:** SRG-APP-000026-CTR-000070
**Rule ID:** SV-242403r712565_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000700
**Rule Title:** Kubernetes API Server must generate audit records that identify what type of event has occurred, identify the source of the event, contain the event results, identify any users, and identify any containers associated with the event.

**Vulnerability Discussion:** Within Kubernetes, audit data for all components is generated by the API server. This audit data is important when there are issues, to include security incidents that must be investigated. To make the audit data worthwhile for the investigation of events, it is necessary to have the appropriate and required data logged. To fully understand the event, it is important to identify any users associated with the event.

The API server policy file allows for the following levels of auditing:
None - Do not log events that match the rule.
Metadata - Log request metadata (requesting user, timestamp, resource, verb, etc.) but not request or response body.
Request - Log event metadata and request body but not response body.
RequestResponse - Log event metadata, request, and response bodies.

Satisfies: SRG-APP-000026-CTR-000070, SRG-APP-000027-CTR-000075, SRG-APP-000028-CTR-000080, SRG-APP-000101-CTR-000205, SRG-APP-000100-CTR-000200, SRG-APP-000100-CTR-000195, SRG-APP-000099-CTR-000190, SRG-APP-000098-CTR-000185, SRG-APP-000095-CTR-000170, SRG-APP-000096-CTR-000175, SRG-APP-000097-CTR-000180, SRG-APP-000507-CTR-001295, SRG-APP-000504-CTR-001280, SRG-APP-000503-CTR-001275, SRG-APP-000501-CTR-001265, SRG-APP-000500-CTR-001260, SRG-APP-000497-CTR-001245, SRG-APP-000496-CTR-001240, SRG-APP-000493-CTR-001225, SRG-APP-000492-CTR-001220, SRG-APP-000343-CTR-000780, SRG-APP-000381-CTR-000905

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i audit-policy-file

If the audit-policy-file is not set, this is a finding.

The file given is the policy file and defines what is audited and what information is included with each event.

The policy file must look like this:

# Log all requests at the RequestResponse level.
apiVersion: audit.k8s.io/vX (Where X is the latest apiVersion)

```
kind: Policy
rules:
- level: RequestResponse
```

If the audit policy file does not look like above, this is a finding.

**Fix Text:** Edit the Kubernetes API Server audit policy and set it to look like the following:

```
# Log all requests at the RequestResponse level.
apiVersion: audit.k8s.io/vX (Where X is the latest apiVersion)
kind: Policy
rules:
- level: RequestResponse
```

**CCI:** CCI-000018

**CCI:** CCI-000130

**CCI:** CCI-000131

**CCI:** CCI-000132

**CCI:** CCI-000133

**CCI:** CCI-000134

**CCI:** CCI-000135

**CCI:** CCI-000172

**CCI:** CCI-001403

**CCI:** CCI-001404

**CCI:** CCI-001487

**CCI:** CCI-001814

**CCI:** CCI-002234

---

**Group ID (Vulid):** V-242404
**Group Title:** SRG-APP-000133-CTR-000290
**Rule ID:** SV-242404r712568_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000850

**Rule Title:** Kubernetes Kubelet must deny hostname override.

**Vulnerability Discussion:** Kubernetes allows for the overriding of hostnames. Allowing this feature to be implemented within the kubelets may break the TLS setup between the kubelet service and the API server. This setting also can make it difficult to associate logs with nodes if security analytics needs to take place. The better practice is to setup nodes with resolvable FQDNs and avoid overriding the hostnames.

**Check Content:**
On the Master and each Worker node, change to the /etc/sysconfig/ directory and run the command:

```
grep -i hostname-override kubelet
--hostname-override
```

If any of the nodes have the setting "hostname-override" present, this is a finding.

**Fix Text:** Edit the Kubernetes Kubelet file in the /etc/sysconfig directory on the Master and Worker nodes and remove the "--hostname-override" setting. Restart the service after the change is made by running:

```
service kubelet restart
```

**CCI:** CCI-001499

---

**Group ID (Vulid):** V-242405
**Group Title:** SRG-APP-000133-CTR-000295
**Rule ID:** SV-242405r712571_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000860
**Rule Title:** The Kubernetes manifests must be owned by root.

**Vulnerability Discussion:** The manifest files contain the runtime configuration of the API server, proxy, scheduler, controller, and etcd. If an attacker can gain access to these files, changes can be made to open vulnerabilities and bypass user authorizations inherit within Kubernetes with RBAC implemented.

**Check Content:**
On the Master node, change to the /etc/kubernetes/manifest directory. Run the command:

```
ls -l *
```

Each manifest file must be owned by root:root.

If any manifest file is not owned by root:root, this is a finding.

**Fix Text:** On the Master node, change to the /etc/kubernetes/manifest directory. Run the command:

```
chown root:root *
```

To verify the change took place, run the command:

```
ls -l *
```

All the manifest files should be owned by root:root.

**CCI:** CCI-001499

---

**Group ID (Vulid):** V-242406
**Group Title:** SRG-APP-000133-CTR-000300
**Rule ID:** SV-242406r712574_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000880
**Rule Title:** The Kubernetes kubelet configuration file must be owned by root.

**Vulnerability Discussion:** The kubelet configuration file contains the runtime configuration of the kubelet service. If an attacker can gain access to this file, changes can be made to open vulnerabilities and bypass user authorizations inherent within Kubernetes with RBAC implemented.

**Check Content:**
On the Master and worker nodes, change to the /etc/sysconfig directory. Run the command:

ls -l kubelet

Each kubelet configuration file must be owned by root:root.

If any manifest file is not owned by root:root, this is a finding.

**Fix Text:** On the Master and Worker nodes, change to the /etc/sysconfig directory. Run the command:

chown root:root kubelet

To verify the change took place, run the command:

ls -l kubelet

The kubelet file should now be owned by root:root.

**CCI:** CCI-001499

---

**Group ID (Vulid):** V-242407
**Group Title:** SRG-APP-000133-CTR-000305
**Rule ID:** SV-242407r799982_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000890
**Rule Title:** The Kubernetes kubelet configuration files must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The kubelet configuration file contains the runtime configuration of the kubelet service. If an attacker can gain access to this file, changes can be made to open vulnerabilities and bypass user authorizations inherit within Kubernetes with RBAC implemented.

**Check Content:**
On the Master and worker nodes, change to the /etc/kubernetes/manifest directory. Run the command:

ls -l kubelet

Each kubelet configuration file must have permissions of "644" or more restrictive.

If any kubelet configuration file is less restrictive than "644", this is a finding.

**Fix Text:** On the Master node, change to the /etc/kubernetes/manifest directory. Run the command:

chmod 644 kubelet

To verify the change took place, run the command:

ls -l kubelet

The kubelet file should now have the permissions of "644".

**CCI:** CCI-001499

---

**Group ID (Vulid):** V-242408
**Group Title:** SRG-APP-000133-CTR-000310
**Rule ID:** SV-242408r712580_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000900
**Rule Title:** The Kubernetes manifests must have least privileges.

**Vulnerability Discussion:** The manifest files contain the runtime configuration of the API server, scheduler, controller, and etcd. If an attacker can gain access to these files, changes can be made to open vulnerabilities and bypass user authorizations inherent within Kubernetes with RBAC implemented.

Satisfies: SRG-APP-000133-CTR-000310, SRG-APP-000133-CTR-000295

**Check Content:**
On the Master node, change to the /etc/kubernetes/manifest directory. Run the command:

ls -l *

Each manifest file must have permissions "644" or more restrictive.

If any manifest file is less restrictive than "644", this is a finding.

**Fix Text:** On the Master node, change to the /etc/kubernetes/manifest directory. Run the command:

chmod 644 *

To verify the change took place, run the command:

ls -l *

All the manifest files should now have privileges of "644".

**CCI:** CCI-001499

---

**Group ID (Vulid):** V-242409
**Group Title:** SRG-APP-000141-CTR-000315

**Rule ID:** SV-242409r712583_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000910
**Rule Title:** Kubernetes Controller Manager must disable profiling.

**Vulnerability Discussion:** Kubernetes profiling provides the ability to analyze and troubleshoot Controller Manager events over a web interface on a host port. Enabling this service can expose details about the Kubernetes architecture. This service must not be enabled unless deemed necessary.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep -i profiling *
```

If the setting "profiling" is not configured in the Kubernetes Controller Manager manifest file or it is set to "True", this is a finding.

**Fix Text:** Edit the Kubernetes Controller Manager manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the argument "--profiling value" to "false".

**CCI:** CCI-000381

---

**Group ID (Vulid):** V-242410
**Group Title:** SRG-APP-000142-CTR-000325
**Rule ID:** SV-242410r808576_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000920
**Rule Title:** The Kubernetes API Server must enforce ports, protocols, and services (PPS) that adhere to the Ports, Protocols, and Services Management Category Assurance List (PPSM CAL).

**Vulnerability Discussion:** Kubernetes API Server PPSs must be controlled and conform to the PPSM CAL. Those PPS that fall outside the PPSM CAL must be blocked. Instructions on the PPSM can be found in DoD Instruction 8551.01 Policy.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep kube-apiserver.manifest -I -secure-port *
grep kube-apiserver.manifest -I -etcd-servers *
-edit manifest file:
VIM <Manifest Name>
Review livenessProbe:
HttpGet:
Port:
Review ports:
- containerPort:
hostPort:
- containerPort:
hostPort:
```

Run Command:
kubectl describe services –all-namespace

Search labels for any apiserver names spaces.
Port:

Any manifest and namespace PPS or services configuration not in compliance with PPSM CAL is a finding.

Review the information systems documentation and interview the team, gain an understanding of the API Server architecture, and determine applicable PPS. If there are any ports, protocols, and services in the system documentation not in compliance with the CAL PPSM, this is a finding. Any PPS not set in the system documentation is a finding.

Review findings against the most recent PPSM CAL:
https://cyber.mil/ppsm/cal/

Verify API Server network boundary with the PPS associated with the CAL Assurance Categories. Any PPS not in compliance with the CAL Assurance Category requirements is a finding.

**Fix Text:** Amend any system documentation requiring revision. Update Kubernetes API Server manifest and namespace PPS configuration to comply with PPSM CAL.

**CCI:** CCI-000382

---

**Group ID (Vulid):** V-242411
**Group Title:** SRG-APP-000142-CTR-000325
**Rule ID:** SV-242411r712589_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000930
**Rule Title:** The Kubernetes Scheduler must enforce ports, protocols, and services (PPS) that adhere to the Ports, Protocols, and Services Management Category Assurance List (PPSM CAL).

**Vulnerability Discussion:** Kubernetes Scheduler PPS must be controlled and conform to the PPSM CAL. Those ports, protocols, and services that fall outside the PPSM CAL must be blocked. Instructions on the PPSM can be found in DoD Instruction 8551.01 Policy.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:
grep kube-scheduler.manifest -I -insecure-port
grep kube-scheduler.manifest -I -secure-port
-edit manifest file:
VIM <Manifest Name>
Review livenessProbe:
HttpGet:
Port:
Review ports:
- containerPort:
hostPort:
- containerPort:
hostPort:
Run Command:
kubectl describe services –all-namespace
Search labels for any scheduler names spaces.
Port:

Any manifest and namespace PPS configuration not in compliance with PPSM CAL is a finding.

Review the information systems documentation and interview the team, gain an understanding of the Scheduler architecture, and determine applicable PPS. Any PPS in the system documentation not in compliance with the CAL PPSM is a finding. Any PPSs not set in the system documentation is a finding.

Review findings against the most recent PPSM CAL:
https://cyber.mil/ppsm/cal/

Verify Scheduler network boundary with the PPS associated with the CAL Assurance Categories. Any PPS not in compliance with the CAL Assurance Category requirements is a finding.

**Fix Text:** Amend any system documentation requiring revision. Update Kubernetes Scheduler manifest and namespace PPS configuration to comply with the PPSM CAL.

**CCI:** CCI-000382

---

**Group ID (Vulid):** V-242412
**Group Title:** SRG-APP-000142-CTR-000330
**Rule ID:** SV-242412r808578_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000940
**Rule Title:** The Kubernetes Controllers must enforce ports, protocols, and services (PPS) that adhere to the Ports, Protocols, and Services Management Category Assurance List (PPSM CAL).

**Vulnerability Discussion:** Kubernetes Controller ports, protocols, and services must be controlled and conform to the PPSM CAL. Those PPS that fall outside the PPSM CAL must be blocked. Instructions on the PPSM can be found in DoD Instruction 8551.01 Policy.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep kube-scheduler.manifest -I -secure-port
-edit manifest file:
VIM <Manifest Name:
Review livenessProbe:
HttpGet:
Port:
Review ports:
- containerPort:
hostPort:
- containerPort:
hostPort:
Run Command:
kubectl describe services –all-namespace
Search labels for any controller names spaces.
```

Any manifest and namespace PPS or services configuration not in compliance with PPSM CAL is a finding.

Review the information systems documentation and interview the team, gain an understanding of the Controller architecture, and determine applicable PPS. Any PPS in the system documentation not in compliance with the CAL PPSM is a finding. Any PPS not set in the system documentation is a finding.

Review findings against the most recent PPSM CAL:

https://cyber.mil/ppsm/cal/

Verify Controller network boundary with the PPS associated with the Controller for Assurance Categories. Any PPS not in compliance with the CAL Assurance Category requirements is a finding.

**Fix Text:** Amend any system documentation requiring revision. Update Kubernetes Controller manifest and namespace PPS configuration to comply with PPSM CAL.

**CCI:** CCI-000382

---

**Group ID (Vulid):** V-242413
**Group Title:** SRG-APP-000142-CTR-000325
**Rule ID:** SV-242413r712595_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000950
**Rule Title:** The Kubernetes etcd must enforce ports, protocols, and services (PPS) that adhere to the Ports, Protocols, and Services Management Category Assurance List (PPSM CAL).

**Vulnerability Discussion:** Kubernetes etcd PPS must be controlled and conform to the PPSM CAL. Those PPS that fall outside the PPSM CAL must be blocked. Instructions on the PPSM can be found in DoD Instruction 8551.01 Policy.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:
grep kube-apiserver.manifest -I -etcd-servers *
-edit etcd-main.manifest file:
VIM <Manifest Name:
Review livenessProbe:
HttpGet:
Port:
Review ports:
- containerPort:
hostPort:
- containerPort:
hostPort:
Run Command:
kubectl describe services –all-namespace
Search labels for any apiserver names spaces.
Port:

Any manifest and namespace PPS configuration not in compliance with PPSM CAL is a finding.

Review the information systems documentation and interview the team, gain an understanding of the etcd architecture, and determine applicable PPS. Any PPS in the system documentation not in compliance with the CAL PPSM is a finding. Any PPS not set in the system documentation is a finding.

Review findings against the most recent PPSM CAL:
https://cyber.mil/ppsm/cal/

Verify etcd network boundary with the PPS associated with the CAL Assurance Categories. Any PPS not in compliance with the CAL Assurance Category requirements is a finding.

**Fix Text:** Amend any system documentation requiring revision. Update Kubernetes etcd manifest and namespace

PPS configuration to comply with PPSM CAL.

**CCI:** CCI-000382

---

**Group ID (Vulid):** V-242414
**Group Title:** SRG-APP-000142-CTR-000330
**Rule ID:** SV-242414r717030_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-000960
**Rule Title:** The Kubernetes cluster must use non-privileged host ports for user pods.

**Vulnerability Discussion:** Privileged ports are those ports below 1024 and that require system privileges for their use. If containers can use these ports, the container must be run as a privileged user. Kubernetes must stop containers that try to map to these ports directly. Allowing non-privileged ports to be mapped to the container-privileged port is the allowable method when a certain port is needed. An example is mapping port 8080 externally to port 80 in the container.

**Check Content:**
On the Master node, run the command:

kubectl get pods --all-namespaces

The list returned is all pods running within the Kubernetes cluster. For those pods running within the user namespaces (System namespaces are kube-system, kube-node-lease and kube-public), run the command:

kubectl get pod podname -o yaml | grep -i port

Note: In the above command, "podname" is the name of the pod. For the command to work correctly, the current context must be changed to the namespace for the pod. The command to do this is:

kubectl config set-context --current --namespace=namespace-name
(Note: "namespace-name" is the name of the namespace.)

Review the ports that are returned for the pod.

If any host-privileged ports are returned for any of the pods, this is a finding.

**Fix Text:** For any of the pods that are using host-privileged ports, reconfigure the pod to use a service to map a host non-privileged port to the pod port or reconfigure the image to use non-privileged ports.

**CCI:** CCI-000382

---

**Group ID (Vulid):** V-242415
**Group Title:** SRG-APP-000171-CTR-000435
**Rule ID:** SV-242415r712601_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-001160
**Rule Title:** Secrets in Kubernetes must not be stored as environment variables.

**Vulnerability Discussion:** Secrets, such as passwords, keys, tokens, and certificates should not be stored as environment variables. These environment variables are accessible inside Kubernetes by the "Get Pod" API call, and by any system, such as CI/CD pipeline, which has access to the definition file of the container. Secrets must be mounted from files or stored within password vaults.

**Check Content:**
On the Kubernetes Master node, run the following command:

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

If any of the values returned reference environment variables, this is a finding.

**Fix Text:** Any secrets stored as environment variables must be moved to the secret files with the proper protections and enforcements or placed within a password vault.

**CCI:** CCI-000196

---

**Group ID (Vulid):** V-245541
**Group Title:** SRG-APP-000190-CTR-000500
**Rule ID:** SV-245541r754888_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001300
**Rule Title:** Kubernetes Kubelet must not disable timeouts.

**Vulnerability Discussion:** Idle connections from the Kubelet can be use by unauthorized users to perform malicious activity to the nodes, pods, containers, and cluster within the Kubernetes Control Plane. Setting the streaming connection idle timeout defines the maximum time an idle session is permitted prior to disconnect. Setting the value to "0" never disconnects any idle sessions. Idle timeouts must never be set to "0" and should be defined at a minimum of "5 minutes".

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node. Run the command:

```
grep -i streaming-connection-idle-timeout kubelet
```

If the setting streaming-connection-idle-timeout is set to "0" or the parameter is not configured in the Kubernetes Kubelet, this is a finding.

**Fix Text:** Edit the Kubernetes Kubelet file in the /etc/sysconfig directory on the Kubernetes Master Node. Set the argument "--streaming-connection-idle-timeout" to a value other than "0". Reset Kubelet service using the following command:

```
service kubelet restart
```

**CCI:** CCI-001133

---

**Group ID (Vulid):** V-242417
**Group Title:** SRG-APP-000211-CTR-000530
**Rule ID:** SV-242417r712607_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001360
**Rule Title:** Kubernetes must separate user functionality.

**Vulnerability Discussion:** Separating user functionality from management functionality is a requirement for all the components within the Kubernetes Control Plane. Without the separation, users may have access to management functions that can degrade the Kubernetes architecture and the services being offered, and can offer a method to bypass testing and validation of functions before introduced into a production environment.

**Check Content:**
On the Master node, run the command:

```
kubectl get pods --all-namespaces
```

Review the namespaces and pods that are returned. Kubernetes system namespaces are kube-node-lease, kube-public, and kube-system.

If any user pods are present in the Kubernetes system namespaces, this is a finding.

**Fix Text:** Move any user pods that are present in the Kubernetes system namespaces to user specific namespaces.

**CCI:** CCI-001082

---

**Group ID (Vulid):** V-242418
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242418r799985_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001400
**Rule Title:** The Kubernetes API server must use approved cipher suites.

**Vulnerability Discussion:** The Kubernetes API server communicates to the kubelet service on the nodes to deploy, update, and delete resources. If an attacker were able to get between this communication and modify the request, the Kubernetes cluster could be compromised. Using approved cypher suites for the communication ensures the protection of the transmitted information, confidentiality, and integrity so that the attacker cannot read or alter this communication.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep -i tls-cipher-suites *
```

If the setting feature tls-cipher-suites is not set in the Kubernetes API server manifest file or contains no value or does not contain
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of tls-cipher-suites to:
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242419
**Group Title:** SRG-APP-000219-CTR-000550

**Rule ID:** SV-242419r712613_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001410
**Rule Title:** Kubernetes API Server must have the SSL Certificate Authority set.

**Vulnerability Discussion:** Kubernetes control plane and external communication is managed by API Server. The main implementation of the API Server is to manage hardware resources for pods and containers using horizontal or vertical scaling. Anyone who can access the API Server can effectively control the Kubernetes architecture. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for API Server, the parameter etcd-cafile must be set. This parameter gives the location of the SSL Certificate Authority file used to secure API Server communication.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep -i client-ca-file *
```

If the setting feature client-ca-file is not set in the Kubernetes API server manifest file or contains no value, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of client-ca-file to path containing Approved Organizational Certificate.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242420
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242420r799987_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001420
**Rule Title:** Kubernetes Kubelet must have the SSL Certificate Authority set.

**Vulnerability Discussion:** Kubernetes container and pod configuration are maintained by Kubelet. Kubelet agents register nodes with the API Server, mount volume storage, and perform health checks for containers and pods. Anyone who gains access to Kubelet agents can effectively control applications within the pods and containers. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for Kubelet, the client-ca-file must be set. This parameter gives the location of the SSL Certificate Authority file used to secure Kubelet communication.

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node. Run the command:

grep -i client-ca-file kubelet

If the setting client-ca-file is not set in the Kubernetes API server manifest file or contains no value, this is a finding.

**Fix Text:** Edit the Kubernetes Kubelet file in the /etc/sysconfig/ directory on the Kubernetes Master Node. Set the value of client-ca-file to path containing Approved Organizational Certificate.

Reset Kubelet service using the following command:
service kubelet restart

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242421
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242421r717033_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001430
**Rule Title:** Kubernetes Controller Manager must have the SSL Certificate Authority set.

**Vulnerability Discussion:** The Kubernetes Controller Manager is responsible for creating service accounts and tokens for the API Server, maintaining the correct number of pods for every replication controller and provides notifications when nodes are offline.

Anyone who gains access to the Controller Manager can generate backdoor accounts, take possession of, or diminish system performance without detection by disabling system notification. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes Controller Manager with a means to be able to authenticate sessions and encrypt traffic.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

grep -i root-ca-file *

If the setting client-ca-file is not set in the Kubernetes Controller Manager manifest file or contains no value, this is a finding.

**Fix Text:** Edit the Kubernetes Controller Manager manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of root-ca-file to path containing Approved Organizational Certificate.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242422
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242422r712622_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001440
**Rule Title:** Kubernetes API Server must have a certificate for communication.

**Vulnerability Discussion:** Kubernetes control plane and external communication is managed by API Server. The main implementation of the API Server is to manage hardware resources for pods and container using horizontal or vertical scaling. Anyone who can access the API Server can effectively control the Kubernetes architecture. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for API Server, the parameter etcd-cafile must be set. This parameter gives the location of the SSL Certificate Authority file used to secure API Server communication.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep -i tls-cert-file *
grep -i tls-private-key-file *
```

If the setting tls-cert-file and private-key-file is not set in the Kubernetes API server manifest file or contains no value, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of tls-cert-file and tls-private-key-file to path containing Approved Organizational Certificate.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242423
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242423r808580_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001450
**Rule Title:** Kubernetes etcd must enable client authentication to secure service.

**Vulnerability Discussion:** Kubernetes container and pod configuration are maintained by Kubelet. Kubelet agents register nodes with the API Server, mount volume storage, and perform health checks for containers and pods. Anyone who gains access to Kubelet agents can effectively control applications within the pods and containers. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for Kubelet, the parameter client-cert-auth must be set. This parameter gives the location of the SSL Certificate Authority file used to secure Kubelet communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i client-cert-auth *
```

If the setting client-cert-auth is not configured in the Kubernetes etcd manifest file or set to "false", this is a finding.

**Fix Text:** Edit the Kubernetes etcd manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--client-cert-auth" to "true" for the etcd.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242424
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242424r799988_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001460
**Rule Title:** Kubernetes Kubelet must enable tls-private-key-file for client authentication to secure service.

**Vulnerability Discussion:** Kubernetes container and pod configuration are maintained by Kubelet. Kubelet agents register nodes with the API Server, mount volume storage, and perform health checks for containers and pods. Anyone who gains access to Kubelet agents can effectively control applications within the pods and containers. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for Kubelet, the tls-private-key-file must be set. This parameter gives the location of the SSL Certificate Authority file used to secure Kubelet communication.

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node. Run the commands:

```
grep -i tls-private-key-file kubelet
```

If the setting "tls-private-key-file" is not configured in the Kubernetes Kubelet, this is a finding.

**Fix Text:** Edit the Kubernetes Kuberlet file in the /etc/sysconfig directory on the Kubernetes Master Node. Set the argument tls-private-key-file to an Approved Organization Certificate. Reset Kubelet service using the following command:

```
service kubelet restart
```

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242425
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242425r712631_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001470
**Rule Title:** Kubernetes Kubelet must enable tls-cert-file for client authentication to secure service.

**Vulnerability Discussion:** Kubernetes container and pod configuration are maintained by Kubelet. Kubelet agents register nodes with the API Server, mount volume storage, and perform health checks for containers and pods. Anyone who gains access to Kubelet agents can effectively control applications within the pods and containers. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for Kubelet, the parameter etcd-cafile must be set. This parameter gives the location of the SSL Certificate Authority file used to secure Kubelet communication.

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node. Run the commands:

grep -i tls-cert-file kubelet

If the setting "tls-cert-file" is not configured in the Kubernetes Kubelet, this is a finding.

**Fix Text:** Edit the Kubernetes Kuberlet file in the /etc/sysconfig directory on the Kubernetes Master Node. Set the argument "tls-cert-file" to an Approved Organization Certificate. Reset Kubelet service using the following command:

service kubelet restart

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242426
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242426r754813_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001480
**Rule Title:** Kubernetes etcd must enable client authentication to secure service.

**Vulnerability Discussion:** Kubernetes container and pod configuration are maintained by Kubelet. Kubelet agents register nodes with the API Server, mount volume storage, and perform health checks for containers and pods. Anyone who gains access to Kubelet agents can effectively control applications within the pods and containers. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for Kubelet, the parameter etcd-cafile must be set. This parameter gives the location of the SSL Certificate Authority file used to secure Kubelet communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i peer-client-cert-auth *

If the setting peer-client-cert-auth is not configured in the Kubernetes etcd manifest file or set to "false", this is a finding.

**Fix Text:** Edit the Kubernetes etcd file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--peer-client-cert-auth" to "true" for the etcd.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242427
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242427r808583_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001490
**Rule Title:** Kubernetes etcd must have a key file for secure communication.

**Vulnerability Discussion:** Kubernetes stores configuration and state information in a distributed key-value store called etcd. Anyone who can write to etcd can effectively control the Kubernetes cluster. Even just reading the contents of etcd could easily provide helpful hints to a would-be attacker. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server and etcd with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for etcd, the parameter key-file must be set. This parameter gives the location of the key file used to secure etcd communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Run the command:
grep -i key-file *

If the setting "key-file" is not configured in the etcd manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes etcd manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--key-file" to the Approved Organizational Certificate.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242428
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242428r808586_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001500
**Rule Title:** Kubernetes etcd must have a certificate for communication.

**Vulnerability Discussion:** Kubernetes stores configuration and state information in a distributed key-value store called etcd. Anyone who can write to etcd can effectively control a Kubernetes cluster. Even just reading the contents of etcd could easily provide helpful hints to a would-be attacker. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server and etcd with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for etcd, the parameter cert-file must be set. This parameter gives the location of the SSL certification file used to secure etcd communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i cert-file *
```

If the setting "cert-file" is not configured in the Kubernetes etcd manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes etcd manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--cert-file" to the Approved Organizational Certificate.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242429
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242429r808589_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001510
**Rule Title:** Kubernetes etcd must have the SSL Certificate Authority set.

**Vulnerability Discussion:** Kubernetes stores configuration and state information in a distributed key-value store called etcd. Anyone who can write to etcd can effectively control a Kubernetes cluster. Even just reading the contents of etcd could easily provide helpful hints to a would-be attacker. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server and etcd with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for etcd, the parameter etcd-cafile must be set. This parameter gives the location of the SSL Certificate Authority file used to secure etcd communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i etcd-cafile *
```

If the setting "etcd-cafile" is not configured in the Kubernetes kube-apiserver manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes kube-apiserver manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--etcd-cafile" to the Certificate Authority for etcd.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242430
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242430r808592_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001520
**Rule Title:** Kubernetes etcd must have a certificate for communication.

**Vulnerability Discussion:** Kubernetes stores configuration and state information in a distributed key-value store called etcd. Anyone who can write to etcd can effectively control your Kubernetes cluster. Even just reading the contents of etcd could easily provide helpful hints to a would-be attacker. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server and etcd with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for etcd, the parameter etcd-certfile must be set. This parameter gives the location of the SSL certification file used to secure etcd communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i etcd-certfile *

If the setting "etcd-certfile" is not set in the Kubernetes kube-apiserver manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes kube-apiserver manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--etcd-certfile" to the certificate to be used for communication with etcd.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242431
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242431r808595_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001530
**Rule Title:** Kubernetes etcd must have a key file for secure communication.

**Vulnerability Discussion:** Kubernetes stores configuration and state information in a distributed key-value store called etcd. Anyone who can write to etcd can effectively control a Kubernetes cluster. Even just reading the contents of etcd could easily provide helpful hints to a would-be attacker. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server and etcd with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for etcd, the parameter etcd-keyfile must be set. This parameter gives the location of the key file used to secure etcd communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i etcd-keyfile *

If the setting "etcd-keyfile" is not configured in the Kubernetes kube-apiserver manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes kube-apiserver manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--etcd-keyfile" to the certificate to be used for communication with etcd.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242432
**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242432r808597_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001540
**Rule Title:** Kubernetes etcd must have peer-cert-file set for secure communication.

**Vulnerability Discussion:** Kubernetes stores configuration and state information in a distributed key-value store called etcd. Anyone who can write to etcd can effectively control the Kubernetes cluster. Even just reading the contents of etcd could easily provide helpful hints to a would-be attacker. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server and etcd with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for etcd, the parameter peer-cert-file must be set. This parameter gives the location of the SSL certification file used to secure etcd communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i peer-cert-file *

If the setting "peer-cert-file" is not configured in the Kubernetes etcd manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes etcd manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--peer-cert-file" to the certificate to be used for communication with etcd.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242433

**Group Title:** SRG-APP-000219-CTR-000550
**Rule ID:** SV-242433r808599_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-001550
**Rule Title:** Kubernetes etcd must have a peer-key-file set for secure communication.

**Vulnerability Discussion:** Kubernetes stores configuration and state information in a distributed key-value store called etcd. Anyone who can write to etcd can effectively control a Kubernetes cluster. Even just reading the contents of etcd could easily provide helpful hints to a would-be attacker. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server and etcd with a means to be able to authenticate sessions and encrypt traffic.

To enable encrypted communication for etcd, the parameter peer-key-file must be set. This parameter gives the location of the SSL certification file used to secure etcd communication.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i peer-key-file *

If the setting "peer-key-file" is not set in the Kubernetes etcd manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes etcd manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node.

Set the value of "--peer-key-file" to the certificate to be used for communication with etcd.

**CCI:** CCI-001184

---

**Group ID (Vulid):** V-242434
**Group Title:** SRG-APP-000233-CTR-000585
**Rule ID:** SV-242434r712658_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-001620
**Rule Title:** Kubernetes Kubelet must enable kernel protection.

**Vulnerability Discussion:** System kernel is responsible for memory, disk, and task management. The kernel provides a gateway between the system hardware and software. Kubernetes requires kernel access to allocate resources to the Control Plane. Threat actors that penetrate the system kernel can inject malicious code or hijack the Kubernetes architecture. It is vital to implement protections through Kubernetes components to reduce the attack surface.

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node. Run the command:

grep -i protect-kernel-defaults kubelet

If the setting "protect-kernel-defaults" is set to false or not set in the Kubernetes Kubelet, this is a finding.

**Fix Text:** Edit the Kubernetes Kuberlet file in the /etc/sysconfig directory on the Kubernetes Master Node. Set the argument "--protect-kernel-defaults" to "true".

Reset Kubelet service using the following command:

service kubelet restart

**CCI:** CCI-001084

---

**Group ID (Vulid):** V-242435
**Group Title:** SRG-APP-000340-CTR-000770
**Rule ID:** SV-242435r712661_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-001990
**Rule Title:** Kubernetes must prevent non-privileged users from executing privileged functions to include disabling, circumventing, or altering implemented security safeguards/countermeasures or the installation of patches and updates.

**Vulnerability Discussion:** Kubernetes uses the API Server to control communication to the other services that makeup Kubernetes. The use of authorizations and not the default of "AlwaysAllow" enables the Kubernetes functions control to only the groups that need them.

To control access the API server must have one of the following options set for the authorization mode:
--authorization-mode=ABAC Attribute-Based Access Control (ABAC) mode allows a user to configure policies using local files.
--authorization-mode=RBAC Role-based access control (RBAC) mode allows a user to create and store policies using the Kubernetes API.
--authorization-mode=Webhook

WebHook is an HTTP callback mode that allows a user to manage authorization using a remote REST endpoint.
--authorization-mode=Node

Node authorization is a special-purpose authorization mode that specifically authorizes API requests made by kubelets.
--authorization-mode=AlwaysDeny

This flag blocks all requests. Use this flag only for testing.

Satisfies: SRG-APP-000340-CTR-000770, SRG-APP-000033-CTR-000095, SRG-APP-000378-CTR-000880

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i authorization-mode *

If the setting authorization-mode is set to "AlwaysAllow" in the Kubernetes API Server manifest file or is not configured, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the argument "--authorization-mode" to any valid authorization mode other than AlwaysAllow.

**CCI:** CCI-000213

**CCI:** CCI-001812

**CCI:** CCI-002235

---

**Group ID (Vulid):** V-242436
**Group Title:** SRG-APP-000342-CTR-000775
**Rule ID:** SV-242436r712664_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-002000
**Rule Title:** The Kubernetes API server must have the ValidatingAdmissionWebhook enabled.

**Vulnerability Discussion:** Enabling the admissions webhook allows for Kubernetes to apply policies against objects that are to be created, read, updated, or deleted. By applying a pod security policy, control can be given to not allow images to be instantiated that run as the root user. If pods run as the root user, the pod then has root privileges to the host system and all the resources it has. An attacker can use this to attack the Kubernetes cluster. By implementing a policy that does not allow root or privileged pods, the pod users are limited in what the pod can do and access.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i ValidatingAdmissionWebhook *
```

If a line is not returned that includes enable-admission-plugins and ValidatingAdmissionWebhook, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the argument "--enable-admission-plugins" to include "ValidatingAdmissionWebhook". Each enabled plugin is separated by commas.

Note: It is best to implement policies first and then enable the webhook, otherwise a denial of service may occur.

**CCI:** CCI-002233

---

**Group ID (Vulid):** V-242437
**Group Title:** SRG-APP-000342-CTR-000775
**Rule ID:** SV-242437r808601_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-002010
**Rule Title:** Kubernetes must have a pod security policy set.

**Vulnerability Discussion:** Enabling the admissions webhook allows for Kubernetes to apply policies against objects that are to be created, read, updated, or deleted. By applying a pod security policy, control can be given to not allow images to be instantiated that run as the root user. If pods run as the root user, the pod then has root privileges to the host system and all the resources it has. An attacker can use this to attack the Kubernetes cluster. By implementing a policy that does not allow root or privileged pods, the pod users are limited in what the pod can do and access.

**Check Content:**

On the Master Node, run the command:

```
kubectl get podsecuritypolicy
```

If there is no pod security policy configured, this is a finding.

For any pod security policies listed, edit the policy with the command:

```
kubectl edit podsecuritypolicy policyname
```
(Note: "policyname" is the name of the policy.)

Review the runAsUser, supplementalGroups and fsGroup sections of the policy.

If any of these sections are missing, this is a finding.

If the rule within the runAsUser section is not set to "MustRunAsNonRoot", this is a finding.

If the ranges within the supplementalGroups section has min set to "0" or min is missing, this is a finding.

If the ranges within the fsGroup section has a min set to "0" or the min is missing, this is a finding.

**Fix Text:** From the Master node, save the following policy to a file called restricted.yml.

```
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
name: restricted
annotations:
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
```

To implement the policy, run the command:

```
kubectl create -f restricted.yml
```

**CCI:** CCI-002233

---

**Group ID (Vulid):** V-242438
**Group Title:** SRG-APP-000435-CTR-001070
**Rule ID:** SV-242438r754802_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-002600
**Rule Title:** Kubernetes API Server must configure timeouts to limit attack surface.

**Vulnerability Discussion:** Kubernetes API Server request timeouts sets the duration a request stays open before timing out. Since the API Server is the central component in the Kubernetes Control Plane, it is vital to protect this service. If request timeouts were not set, malicious attacks or unwanted activities might affect multiple deployments across different applications or environments. This might deplete all resources from the Kubernetes infrastructure causing the information system to go offline. The request-timeout value must never be set to "0". This disables the request-timeout feature. By default, the request-timeout is set to "1 minute".

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep -I request-timeout *
```

If Kubernetes API Server manifest file does not exist, this is a finding.
If the setting request-timeout is set to "0" in the Kubernetes API Server manifest file, or is not configured this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of request-timeout greater than "0".

**CCI:** CCI-002385

---

**Group ID (Vulid):** V-245542

**Group Title:** SRG-APP-000439-CTR-001080
**Rule ID:** SV-245542r754891_rule
**Severity: CAT I**
**Rule Version (STIG-ID):** CNTR-K8-002620
**Rule Title:** Kubernetes API Server must disable basic authentication to protect information in transit.

**Vulnerability Discussion:** Kubernetes basic authentication sends and receives request containing username, uid, groups, and other fields over a clear text HTTP communication. Basic authentication does not provide any security mechanisms using encryption standards. PKI certificate-based authentication must be set over a secure channel to ensure confidentiality and integrity. Basic authentication must not be set in the manifest file.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

grep -i basic-auth-file *

If "basic-auth-file" is set in the Kubernetes API server manifest file this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Remove the setting "--basic-auth-file".

**CCI:** CCI-002418

---

**Group ID (Vulid):** V-245543
**Group Title:** SRG-APP-000439-CTR-001080
**Rule ID:** SV-245543r754894_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-002630
**Rule Title:** Kubernetes API Server must disable token authentication to protect information in transit.

**Vulnerability Discussion:** Kubernetes token authentication uses password known as secrets in a plaintext file. This file contains sensitive information such as token, username and user uid. This token is used by service accounts within pods to authenticate with the API Server. This information is very valuable for attackers with malicious intent if the service account is privileged having access to the token. With this token a threat actor can impersonate the service account gaining access to the Rest API service.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

grep -i token-auth-file *

If "token-auth-file" is set in the Kubernetes API server manifest file, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Remove parameter "--token-auth-file".

**CCI:** CCI-002418

---

**Group ID (Vulid):** V-245544
**Group Title:** SRG-APP-000439-CTR-001080
**Rule ID:** SV-245544r754897_rule

**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-002640
**Rule Title:** Kubernetes endpoints must use approved organizational certificate and key pair to protect information in transit.

**Vulnerability Discussion:** Kubernetes control plane and external communication is managed by API Server. The main implementation of the API Server is to manage hardware resources for pods and container using horizontal or vertical scaling. Anyone who can gain access to the API Server can effectively control your Kubernetes architecture. Using authenticity protection, the communication can be protected against man-in-the-middle attacks/session hijacking and the insertion of false information into sessions.

The communication session is protected by utilizing transport encryption protocols, such as TLS. TLS provides the Kubernetes API Server with a means to be able to authenticate sessions and encrypt traffic.

By default, the API Server does not authenticate to the kubelet HTTPs endpoint. To enable secure communication for API Server, the parameter -kubelet-client-certificate and kubelet-client-key must be set. This parameter gives the location of the certificate and key pair used to secure API Server communication.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep -i kubelet-client-certificate *
grep -I kubelet-client-key *
```

If the setting "--kubelet-client-certificate" is not configured in the Kubernetes API server manifest file or contains no value, this is a finding.

If the setting "--kubelet-client-key" is not configured in the Kubernetes API server manifest file or contains no value, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--kubelet-client-certificate" and "--kubelet-client-key" to an Approved Organizational Certificate and key pair.

**CCI:** CCI-002418

---

**Group ID (Vulid):** V-242442
**Group Title:** SRG-APP-000454-CTR-001110
**Rule ID:** SV-242442r712682_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-002700
**Rule Title:** Kubernetes must remove old components after updated versions have been installed.

**Vulnerability Discussion:** Previous versions of Kubernetes components that are not removed after updates have been installed may be exploited by adversaries by allowing the vulnerabilities to still exist within the cluster. It is important for Kubernetes to remove old pods when newer pods are created using new images to always be at the desired security state.

**Check Content:**
To view all pods and the images used to create the pods, from the Master node, run the following command:

```
kubectl get pods --all-namespaces -o jsonpath="{..image}" | \
tr -s '[[:space:]]' '\n' | \
sort | \
uniq -c
```

Review the images used for pods running within Kubernetes.

If there are multiple versions of the same image, this is a finding.

**Fix Text:** Remove any old pods that are using older images. On the Master node, run the command:

```
kubectl delete pod podname
```

(Note: "podname" is the name of the pod to delete.)

**CCI:** CCI-002617

---

**Group ID (Vulid):** V-242443
**Group Title:** SRG-APP-000456-CTR-001125
**Rule ID:** SV-242443r712685_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-002720
**Rule Title:** Kubernetes must contain the latest updates as authorized by IAVMs, CTOs, DTMs, and STIGs.

**Vulnerability Discussion:** Kubernetes software must stay up to date with the latest patches, service packs, and hot fixes. Not updating the Kubernetes control plane will expose the organization to vulnerabilities.

Flaws discovered during security assessments, continuous monitoring, incident response activities, or information system error handling must also be addressed expeditiously.

Organization-defined time periods for updating security-relevant container platform components may vary based on a variety of factors including, for example, the security category of the information system or the criticality of the update (i.e., severity of the vulnerability related to the discovered flaw).

This requirement will apply to software patch management solutions that are used to install patches across the enclave and also to applications themselves that are not part of that patch management solution. For example, many browsers today provide the capability to install their own patch software. Patch criticality, as well as system criticality will vary. Therefore, the tactical situations regarding the patch management process will also vary. This means that the time period utilized must be a configurable parameter. Time frames for application of security-relevant software updates may be dependent upon the IAVM process.

The container platform components will be configured to check for and install security-relevant software updates within an identified time period from the availability of the update. The container platform registry will ensure the images are current. The specific time period will be defined by an authoritative source (e.g., IAVM, CTOs, DTMs, and STIGs).

**Check Content:**
Authenticate on the Kubernetes Master Node. Run the command:

```
kubectl version --short
```

If kubectl version has a setting not supporting Kubernetes skew policy, this is a finding.

Note: Kubernetes Skew Policy can be found at: https://kubernetes.io/docs/setup/release/version-skew-policy/#supported-versions

**Fix Text:** Upgrade Kubernetes to the supported version. Institute and adhere to the policies and procedures to ensure that patches are consistently applied within the time allowed.

**CCI:** CCI-002605

---

**Group ID (Vulid):** V-242444
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242444r712688_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003110
**Rule Title:** The Kubernetes component manifests must be owned by root.

**Vulnerability Discussion:** The Kubernetes manifests are those files that contain the arguments and settings for the Master Node services. These services are etcd, the api server, controller, proxy, and scheduler. If these files can be changed, the scheduler will be implementing the changes immediately. Many of the security settings within the document are implemented through these manifests.

**Check Content:**
Review the ownership of the Kubernetes manifests files by using the command:

```
stat -c %U:%G /etc/kubernetes/manifests/* | grep -v root:root
```

If the command returns any non root:root file permissions, this is a finding.

**Fix Text:** Change the ownership of the manifest files to root: root by executing the command:

```
chown root:root /etc/kubernetes/manifests/*
```

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242445
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242445r712691_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003120
**Rule Title:** The Kubernetes component etcd must be owned by etcd.

**Vulnerability Discussion:** The Kubernetes etcd key-value store provides a way to store data to the Master Node. If these files can be changed, data to API object and the master node would be compromised. The scheduler will implement the changes immediately. Many of the security settings within the document are implemented through this file.

**Check Content:**
Review the ownership of the Kubernetes etcd files by using the command:

```
stat -c %U:%G /var/lib/etcd/* | grep -v etcd:etcd
```

If the command returns any non etcd:etcd file permissions, this is a finding.

**Fix Text:** Change the ownership of the manifest files to etcd:etcd by executing the command:

chown etcd:etcd /var/lib/etcd/*

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242446
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242446r712694_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003130
**Rule Title:** The Kubernetes conf files must be owned by root.

**Vulnerability Discussion:** The Kubernetes conf files contain the arguments and settings for the Master Node services. These services are controller and scheduler. If these files can be changed, the scheduler will be implementing the changes immediately. Many of the security settings within the document are implemented through this file.

**Check Content:**
Review the Kubernetes conf files by using the command:

```
stat -c %U:%G /etc/kubernetes/admin.conf | grep -v root:root
stat -c %U:%G /etc/kubernetes/scheduler.conf | grep -v root:root
stat -c %U:%G /etc/kubernetes/controller-manager.conf | grep -v root:root
```

If the command returns any non root:root file permissions, this is a finding.

**Fix Text:** Change the ownership of the conf files to root: root by executing the command:

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

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242447
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242447r712697_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003140
**Rule Title:** The Kubernetes Kube Proxy must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes kube proxy kubeconfig contain the argument and setting for the Master Nodes. These settings contain network rules for restricting network communication between pods, clusters, and networks. If these files can be changed, data traversing between the Kubernetes Control Panel components would be compromised. Many of the security settings within the document are implemented through this file.

**Check Content:**
Check if Kube-Proxy is running and obtain --kubeconfig parameter use the following command:
```
ps -ef | grep kube-proxy
```

If Kube-Proxy exists:

Review the permissions of the Kubernetes Kube Proxy by using the command:
stat -c %a <location from --kubeconfig>

If the file has permissions more permissive than "644", this is a finding.

**Fix Text:** Change the permissions of the Kube Proxy to "644" by executing the command:

chown 644 <location from kubeconfig>.

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242448
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242448r712700_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003150
**Rule Title:** The Kubernetes Kube Proxy must be owned by root.

**Vulnerability Discussion:** The Kubernetes kube proxy kubeconfig contain the argument and setting for the Master Nodes. These settings contain network rules for restricting network communication between pods, clusters, and networks. If these files can be changed, data traversing between the Kubernetes Control Panel components would be compromised. Many of the security settings within the document are implemented through this file.

**Check Content:**
Check if Kube-Proxy is running use the following command:
ps -ef | grep kube-proxy

If Kube-Proxy exists:
Review the permissions of the Kubernetes Kube Proxy by using the command:
stat -c %U:%G <location from --kubeconfig>| grep -v root:root

If the command returns any non root:root file permissions, this is a finding.

**Fix Text:** Change the ownership of the Kube Proxy to root:root by executing the command:

chown root:root <location from kubeconfig>.

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242449
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242449r712703_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003160
**Rule Title:** The Kubernetes Kubelet certificate authority file must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes kubelet certificate authority file contains settings for the Kubernetes Node TLS certificate authority. Any request presenting a client certificate signed by one of the authorities in the client-ca-file is authenticated with an identity corresponding to the CommonName of the client certificate. If this file can be changed, the Kubernetes architecture could be compromised. The scheduler will implement the changes immediately. Many of the security settings within the document are implemented through this file.

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node. Run command:

more kubelet
--client-ca-file argument
Note certificate location

If the ca-file argument location file has permissions more permissive than "644", this is a finding.

**Fix Text:** Change the permissions of the --client-ca-file to "644" by executing the command:

chown 644 <kubelet --client--ca-file argument location>.

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242450
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242450r754804_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003170
**Rule Title:** The Kubernetes Kubelet certificate authority must be owned by root.

**Vulnerability Discussion:** The Kubernetes kube proxy kubeconfig contain the argument and setting for the Master Nodes. These settings contain network rules for restricting network communication between pods, clusters, and networks. If these files can be changed, data traversing between the Kubernetes Control Panel components would be compromised. Many of the security settings within the document are implemented through this file.

**Check Content:**
Change to the /etc/sysconfig/ directory on the Kubernetes Master Node.
Review the ownership of the Kubernetes client-ca-file by using the command:
more kubelet
--client-ca-file argument
Note certificate location

Review the ownership of the Kubernetes client-ca-file by using the command:
stat -c %U:%G <location from --client-ca-file argument>| grep -v root:root

If the command returns any non root:root file permissions, this is a finding.

**Fix Text:** Change the permissions of the Kube Proxy to "root" by executing the command:

chown root:root <location from kubeconfig>.

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242451
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242451r712709_rule
**Severity: CAT II**

**Rule Version (STIG-ID):** CNTR-K8-003180
**Rule Title:** The Kubernetes component PKI must be owned by root.

**Vulnerability Discussion:** The Kubernetes PKI directory contains all certificates (.crt files) supporting secure network communications in the Kubernetes Control Plane. If these files can be modified, data traversing within the architecture components would become unsecure and compromised. Many of the security settings within the document are implemented through this file.

**Check Content:**
Review the PKI files in Kubernetes by using the command:

```
ls -laR /etc/kubernetes/pki/
```

If the command returns any non root:root file permissions, this is a finding.

**Fix Text:** Change the ownership of the PKI to root: root by executing the command:

```
chown -R root:root /etc/kubernetes/pki/
```

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242452
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242452r712712_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003190
**Rule Title:** The Kubernetes kubelet config must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes kubelet agent registers nodes with the API Server, mounts volume storage for pods, and performs health checks to containers within pods. If these files can be modified, the information system would be unaware of pod or container degradation. Many of the security settings within the document are implemented through this file.

**Check Content:**
Review the permissions of the Kubernetes Kubelet conf by using the command:

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

If any of the files are have permissions more permissive than "644", this is a finding.

**Fix Text:** Change the permissions of the Kubelet to "644" by executing the command:

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

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242453
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242453r712715_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003200

**Rule Title:** The Kubernetes kubelet config must be owned by root.

**Vulnerability Discussion:** The Kubernetes kubelet agent registers nodes with the API server and performs health checks to containers within pods. If these files can be modified, the information system would be unaware of pod or container degradation. Many of the security settings within the document are implemented through this file.

**Check Content:**
Review the Kubernetes Kubelet conf files by using the command:

stat -c %U:%G /etc/kubernetes/kubelet.conf| grep -v root:root

If the command returns any non root:root file permissions, this is a finding.

**Fix Text:** Change the ownership of the kubelet.conf to root: root by executing the command:

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

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242454
**Group Title:** SRG-APP-000516-CTR-001325
**Rule ID:** SV-242454r754819_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003210
**Rule Title:** The Kubernetes kubeadm.conf must be owned by root.

**Vulnerability Discussion:** The Kubernetes kubeeadm.conf contains sensitive information regarding the cluster nodes configuration. If this file can be modified, the Kubernetes Platform Plane would be degraded or compromised for malicious intent. Many of the security settings within the document are implemented through this file.

**Check Content:**
Review the Kubeadm.conf file :

Get the path for Kubeadm.conf by running:
sytstemctl status kubelet

Note the configuration file installed by the kubeadm is written to
(Default Location: /etc/systemd/system/kubelet.service.d/10-kubeadm.conf)
stat -c %U:%G <kubeadm.conf path> | grep -v root:root

If the command returns any non root:root file permissions, this is a finding.

**Fix Text:** Change the ownership of the kubeadm.conf to root: root by executing the command:

chown root:root <kubeadm.conf path>

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242455
**Group Title:** SRG-APP-000516-CTR-001325

**Rule ID:** SV-242455r754822_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003220
**Rule Title:** The Kubernetes kubeadm.conf must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes kubeadm.conf contains sensitive information regarding the cluster nodes configuration. If this file can be modified, the Kubernetes Platform Plane would be degraded or compromised for malicious intent. Many of the security settings within the document are implemented through this file.

**Check Content:**
Review the kubeadm.conf file :

Get the path for kubeadm.conf by running:
systemctl status kubelet

Note the configuration file installed by the kubeadm is written to
(Default Location: /etc/systemd/system/kubelet.service.d/10-kubeadm.conf)
stat -c %a <kubeadm.conf path>

If the file has permissions more permissive than "644", this is a finding.

**Fix Text:** Change the permissions of kubeadm.conf to "644" by executing the command:

chmod 644 <kubeadm.conf path>

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242456
**Group Title:** SRG-APP-000516-CTR-001330
**Rule ID:** SV-242456r712724_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003230
**Rule Title:** The Kubernetes kubelet config must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes kubelet agent registers nodes with the API server and performs health checks to containers within pods. If this file can be modified, the information system would be unaware of pod or container degradation.

**Check Content:**
Review the permissions of the Kubernetes config.yaml by using the command:

stat -c %a /var/lib/kubelet/config.yaml

If any of the files are have permissions more permissive than "644", this is a finding.

**Fix Text:** Change the permissions of the config.yaml to "644" by executing the command:

chown 644 /var/lib/kubelet/config.yaml

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242457
**Group Title:** SRG-APP-000516-CTR-001330
**Rule ID:** SV-242457r712727_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003240
**Rule Title:** The Kubernetes kubelet config must be owned by root.

**Vulnerability Discussion:** The Kubernetes kubelet agent registers nodes with the API Server and performs health checks to containers within pods. If this file can be modified, the information system would be unaware of pod or container degradation.

**Check Content:**
Review the Kubernetes Kubeadm kubelet conf file by using the command:

```
stat -c %U:%G /var/lib/kubelet/config.yaml| grep -v root:root
```

If the command returns any non root:root file permissions, this is a finding.

**Fix Text:** Change the ownership of the kubelet config to "root: root" by executing the command:

```
chown root:root /var/lib/kubelet/config.yaml
```

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242458
**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242458r754806_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003250
**Rule Title:** The Kubernetes API Server must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes manifests are those files that contain the arguments and settings for the Master Node services. These services are etcd, the API Server, controller, proxy, and scheduler. If these files can be changed, the scheduler will be implementing the changes immediately. Many of the security settings within the document are implemented through these manifests.

**Check Content:**
Review the permissions of the Kubernetes Kubelet by using the command:

```
stat -c %a /etc/kubernetes/manifests/*
```

If any of the files are have permissions more permissive than "644", this is a finding.

**Fix Text:** Change the permissions of the manifest files by executing the command:

```
chmod 644 /etc/kubernetes/manifests/*
```

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242459

**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242459r712733_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003260
**Rule Title:** The Kubernetes etcd must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes etcd key-value store provides a way to store data to the Master Node. If these files can be changed, data to API object and master node would be compromised.

**Check Content:**
Review the permissions of the Kubernetes etcd by using the command:

```
stat -c %a /var/lib/etcd/*
```

If any of the files are have permissions more permissive than "644", this is a finding.

**Fix Text:** Change the permissions of the manifest files to "644" by executing the command:

```
chmod 644/var/lib/etcd/*
```

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242460
**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242460r712736_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003270
**Rule Title:** The Kubernetes admin.conf must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes conf files contain the arguments and settings for the Master Node services. These services are controller and scheduler. If these files can be changed, the scheduler will be implementing the changes immediately.

**Check Content:**
Review the permissions of the Kubernetes config files by using the command:

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

If any of the files are have permissions more permissive than "644", this is a finding.

**Fix Text:** Change the permissions of the conf files to "644" by executing the command:

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

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242461

**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242461r712739_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003280
**Rule Title:** Kubernetes API Server audit logs must be enabled.

**Vulnerability Discussion:** Kubernetes API Server validates and configures pods and services for the API object. The REST operation provides frontend functionality to the cluster share state. Enabling audit logs provides a way to monitor and identify security risk events or misuse of information. Audit logs are necessary to provide evidence in the case the Kubernetes API Server is compromised requiring a Cyber Security Investigation.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

grep -i audit-policy-file *

If the setting "audit-policy-file" is not set or is found in the Kubernetes API manifest file without valid content, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the argument "--audit-policy-file" to "log file directory".

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242462
**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242462r712742_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003290
**Rule Title:** The Kubernetes API Server must be set to audit log max size.

**Vulnerability Discussion:** The Kubernetes API Server must be set for enough storage to retain log information over the period required. When audit logs are large in size, the monitoring service for events becomes degraded. The function of the maximum log file size is to set these limits.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

grep -i audit-log-maxsize *

If the setting "audit-log-maxsize" is not set in the Kubernetes API Server manifest file or it is set to less than "100", this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of –"--audit-log-maxsize" to a minimum of "100".

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242463
**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242463r712745_rule

**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003300
**Rule Title:** The Kubernetes API Server must be set to audit log maximum backup.

**Vulnerability Discussion:** The Kubernetes API Server must set enough storage to retain logs for monitoring suspicious activity and system misconfiguration, and provide evidence for Cyber Security Investigations.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

grep -i audit-log-maxbackup *

If the setting "audit-log-maxbackup" is not set in the Kubernetes API Server manifest file or it is set less than "10", this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--audit-log-maxbackup" to a minimum of "10".

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242464
**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242464r754808_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003310
**Rule Title:** The Kubernetes API Server audit log retention must be set.

**Vulnerability Discussion:** The Kubernetes API Server must set enough storage to retain logs for monitoring suspicious activity and system misconfiguration, and provide evidence for Cyber Security Investigations.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

grep -i audit-log-maxage *

If the setting "audit-log-maxage" is not set in the Kubernetes API Server manifest file or it is set less than "30", this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--audit-log-maxage" to a minimum of "30".

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242465
**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242465r754810_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003320
**Rule Title:** The Kubernetes API Server audit log path must be set.

**Vulnerability Discussion:** Kubernetes API Server validates and configures pods and services for the API object.

The REST operation provides frontend functionality to the cluster share state. Audit logs are necessary to provide evidence in the case the Kubernetes API Server is compromised requiring Cyber Security Investigation. To record events in the audit log the log path value must be set.

**Check Content:**
Change to the /etc/kubernetes/manifests/ directory on the Kubernetes Master Node. Run the command:

```
grep -i audit-log-path *
```

If the setting audit-log-path is not set in the Kubernetes API Server manifest file or it is not set to a valid path, this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--audit-log-path" to valid location.

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242466
**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242466r712754_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003330
**Rule Title:** The Kubernetes PKI CRT must have file permissions set to 644 or more restrictive.

**Vulnerability Discussion:** The Kubernetes PKI directory contains all certificates (.crt files) supporting secure network communications in the Kubernetes Control Plane. If these files can be modified, data traversing within the architecture components would become unsecure and compromised.

**Check Content:**
Review the permissions of the Kubernetes PKI cert files by using the command:

```
find /etc/kubernetes/pki -name "*.crt" | xargs stat -c '%n %a'
```

If any of the files are have permissions more permissive than "644", this is a finding.

**Fix Text:** Change the ownership of the cert files to "644" by executing the command:

```
chmod -R 644 /etc/kubernetes/pki/*.crt
```

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242467
**Group Title:** SRG-APP-000516-CTR-001335
**Rule ID:** SV-242467r712757_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003340
**Rule Title:** The Kubernetes PKI keys must have file permissions set to 600 or more restrictive.

**Vulnerability Discussion:** The Kubernetes PKI directory contains all certificate key files supporting secure network communications in the Kubernetes Control Plane. If these files can be modified, data traversing within the architecture components would become unsecure and compromised.

**Check Content:**
Review the permissions of the Kubernetes PKI key files by using the command:

```
find /etc/kubernetes/pki -name "*.key" | xargs stat -c '%n %a'
```

If any of the files are have permissions more permissive than "600", this is a finding.

**Fix Text:** Change the ownership of the cert files to "600" by executing the command:

```
chmod -R 600 /etc/kubernetes/pki/*.key
```

**CCI:** CCI-000366

---

**Group ID (Vulid):** V-242468
**Group Title:** SRG-APP-000560-CTR-001340
**Rule ID:** SV-242468r712760_rule
**Severity: CAT II**
**Rule Version (STIG-ID):** CNTR-K8-003350
**Rule Title:** The Kubernetes API Server must prohibit communication using TLS version 1.0 and 1.1, and SSL 2.0 and 3.0.

**Vulnerability Discussion:** The Kubernetes API Server will prohibit the use of SSL and unauthorized versions of TLS protocols to properly secure communication.

The use of unsupported protocol exposes vulnerabilities to Kubernetes by rogue traffic interceptions, man-in-the middle attacks, and impersonation of users or services from the container platform runtime, registry, and keystore. To enable the minimum version of TLS to be used by the Kubernetes API Server, the setting "tls-min-version" must be set.

The container platform and its components will adhere to NIST 800-52R2.

**Check Content:**
Change to the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Run the command:

```
grep -i tls-min-version *
```

If the setting tls-min-version is not configured in the Kubernetes API Server manifest file or it is set to "VersionTLS10" or "VersionTLS11", this is a finding.

**Fix Text:** Edit the Kubernetes API Server manifest file in the /etc/kubernetes/manifests directory on the Kubernetes Master Node. Set the value of "--tls-min-version" to either "VersionTLS12" or higher.

**CCI:** CCI-001453

---

# UNCLASSIFIED