Section	Threat	Best Practice	Kubernetes Specific Information
1	Authentication		
1.1	Unauthenticated access to APIs is provided by the container orchestration tool, allowing unauthorized modification of workloads.	a. All access to orchestration tools components and supporting services–for example, monitoring–from users or other services should be configured to require authentication and individual accountability.	In general for Kubernetes itself there are two APIs which may be available without authentication. The first is the insecure API server, which has been disabled in recent versions, the other is the read-only Kubelet which defaults to disabled in recent versions but is still available. This recommendation is that neither of these is used. With regards to supporting systems it's important to ensure that services like Prometheus are configured to require authentication
1.2	Generic administrator accounts are in place for container orchestration tool management. The use of these accounts would prevent non-repudiation of individuals with administrator account access.	a. All user credentials used to authenticate to the orchestration should be tied to specific individuals. Generic credentials should not be used. When a default account is present and cannot be deleted, changing the default password to a strong unique password and then disabling the account will prevent a malicious individual from re-enabling the account and gaining access with the default password	The main area where generic credentials are found in Kubernetes is the user created when a cluster is first setup. This user will often have cluster-admin rights. It should not be used for day-to-day administration and should be kept offline in a secrets management service for break-glass scenarios. Additionally generic credentials should not be created during a clusters lifetime.

Section	Threat	Best Practice	Kubernetes Specific Information
1.3	Credentials, such as client certificates, do not provide for revocation. Lost credentials present a risk of unauthorized access to cluster APIs.	a. All credentials used by the orchestration system should be revokable.	There are two forms of Kubernetes authentication which lack specific support for revocation of credentials. The first is client certificate authentication which is commonly configured. The second is the JWT tokens provided by the TokenRequest API. From a risk perspective the main problem is client certificates as they typically have long expiries, however in many clusters the Token request API can also issue long lasting irrevocable credentials
1.4	Credentials used to access administrative accounts for either containers or container orchestration tools are stored insecurely, leading to unauthorized access to containers or sensitive data.	a. Authentication mechanisms used by the orchestration system should store credentials in a properly secured datastore.	When Kubernetes is storing credentials for "Static Token File" authentication, these are stored in plain text on control plane server disks, so this should not be used.
1.5	Availability of automatic credentials for any workloads running in the cluster. These credentials are susceptible to abuse, particularly if given excessive rights.	a. Credentials for the orchestration system should only be provided to services running in the cluster where explicitly required. b. Service accounts should be configured for least privilege. The level of rights they will have is dependent on how the cluster RBAC is configured.	With Kubernetes this relates to the default setting to mount service account tokens into all workloads by default. Workloads and service account should be configured to only mount service account tokens where they are required for the operation of the workload. The second requirement generally relates to avoiding excessive permissions granted to service account tokens.
1.6	Static credentials–i.e., passwords–used by administrators or service accounts are susceptible to credential stuffing, phishing, keystroke logging, local discovery, extortion, password spray, and brute force attacks	a. Interactive users accessing container orchestration APIs should use multi-factor authentication (MFA).	Typically to get MFA authentication with a cluster, external autentication services are required, OIDC/Oauth typically would be used. With Cloud managed Kubernetes distributions this can be tied to Cloud IAM services which should be configured to require MFA

Section	Threat	Best Practice	Kubernetes Specific Information
2	Authorization		
2.1	Excessive access rights to the container orchestration API could allow users to modify workloads without authorization.	a. Access granted to orchestration systems for users or services should be on a least privilege basis. Blanket administrative access should not be used.	With kubernetes the in-built clusterroles (e.g. cluster-admin, edit, view) should not be used, instead cluster operators should create custom clusterroles which provide only the access required. Cluster-admin in particular should not be used for general administration work.
2.2	Excessive access rights to the container orchestration tools may be provided through the use of hard-coded access groups	a. All access granted to the orchestration tool should be capable of modification. b. Access groups should not be hard-coded.	For Kubernetes this relates to the group "system:masters". This group should not be used for any administrative or service account it is only required in "break glass" circumstances if the entire RBAC configuration for the cluster has been broken
2.3	Accounts may accumulate permissions without documented approvals.	a. Use manual and automated means to regularly audit implemented permissions	No specific content
3	Workload Security		
3.1	Access to shared resources on the underlying host permits container breakouts to occur, compromising the security of shared resources	a. Workloads running in the orchestration system should be configured to prevent access to the underlying cluster nodes by default. Where granted, any access to resources provided by the nodes should be provided on a least privilege basis, and the use of “privileged” mode containers should be specifically avoided.	By default, pods created in a Kuberntes cluster can easily get privileged access to the underlying host. All clusters should have either Pod Security Admission or a 3rd party admission controller running, implementing either the baseline or restricted Pod Security Standard profiles.

Section	Threat	Best Practice	Kubernetes Specific Information
3.2	The use of non-specific versions of container images could facilitate a supply chain attack where a malicious version of the image is pushed to a registry by an attacker.	a. Workload definitions/manifests should target specific known versions of any container images. This should be done via a reliable mechanism checking the cryptographic signatures of images. If signatures are not available, message-digests should be used.	Ideally all clusters shold use image signing and verification (e.g. Cosign in conjunction with Kyverno) to ensure that expected images are deployed to the cluster. Where signing is not available, specific tags (not :latest) should be used
3.3	Containers retrieved from untrusted sources may contain malware or exploitable vulnerabilities.	a. All container images running in the cluster should come from trusted sources.	General images from Docker Hub (outwith the official images) and other public registries should not be used in production clusters.
4	Network Security		
4.1	Container technologies with container networks that do not support network segmentation or restriction allow unauthorized network access between containers.	a. Container orchestration tool networks should be configured on a default deny basis, with access explicitly required only for the operation of the applications being allowed.	For Kubernetes this requires implementation of network policy, and configuring policies which apply to every workload in the cluster. This is typically done by defining default deny policys for ingress and egress for each namespace, then adding specific allow rules as required for workloads to operate
4.2	Access from the container or other networks to the orchestration component and administrative APIs could allow privilege escalation attacks.	a. Access to orchestration system component and other administrative APIs should be restricted using an explicit allow-list of IP addresses.	For access from inside the cluster this would be covered by network policy. For access from external addresses some form of firewalling should restrict access. For Cloud managed kubernetes (e.g. AKS, EKS, GKE) private cluster features would help address this issue
4.3	Unencrypted traffic with management APIs is allowed as a default setting, allowing packet sniffing or spoofing attacks.	a. All traffic with orchestration system components APIs should be over encrypted connections, ensuring encryption key rotation meets PCI key and secret requirements.	For kubernetes components this is generally done over encrypted connections (apart from the read-only kubelet port and insecure API port)
5	PKI		

Section	Threat	Best Practice	Kubernetes Specific Information
5.1	Inability of some container orchestration tool products to support revocation of certificates may lead to misuse of a stolen or lost certificate by attackers.	a. Where revocation of certificates is not supported, certificate-based authentication should not be used. b. Rotate certificates as required by PCI or customer policies or if any containers are compromised.	Its important to note that certificate revocation is not supported by Kubernetes and as such client certificate authentication should not be used apart from in cases of system component--> system component authentication where there are no available alternatives.
	PKI and Certificate Authority services integrated within container orchestration tools may not provide sufficient security outside of the container orchestration tool environment, which could lead to exploitation of other services that attempt to use this chain of trust.	a. The certificates issued by orchestration tools should not be trusted outside of the container orchestrator environment, as the container orchestrator’s Certificate Authority private key can have weaker protection than other enterprise PKI trust chains.	No specific content
6	Secrets Management		
6.1	Inappropriately stored secrets, including credentials, provided through the container orchestration tool, could be leaked to unauthorized users or attackers with some level of access to the environment.	a. All secrets needed for the operation of applications hosted on the orchestration platform should be held in encrypted dedicated secrets management systems.	For production systems the expectation is that a dedicated secrets management system would be used. In particular no secret information should be stored using Kubernetes ConfigMap objects.
6.2	Secrets stored without version control could lead to an outage if a compromise occurs and there is a requirement to rotate them quickly.	a. Apply version control for secrets, so it is easy to refresh or revoke it in case of a compromise.	No specific content
7	Container Orchestration Tool Auditing		
7.1	Existing inventory management and logging solutions may not suffice due to the ephemeral nature of containers and container orchestration tools integration.	a. Access to the orchestration system API(s) should be audited and monitored for indications of unauthorized access. Audit logs should be securely stored on a centralized system.	For Kubernetes this means that audit loggging should be enabled, and these logs should be stored in a centralized system
8	Container Monitoring		

Section	Threat	Best Practice	Kubernetes Specific Information
8.1	Local logging solutions will not allow for appropriate correlation of security events where containers are regularly destroyed.	a. Centralized logging of container activity should be implemented and allow for correlation of events across instances of the same container.	For Kubernetes this means that container logs should be shipped to a centralized service
8.2	Without appropriate detection facilities, the ephemeral nature of containers may allow attackers to execute attacks unnoticed	a. Controls should be implemented to detect the adding and execution of new binaries and unauthorized modification of container files to running containers.	For Kubernetes this would typically be implemented via agent based intrusion detection systems. It's important to note that these systems should be container aware and not just log information at a host level
9	Container Runtime Security		
9.1	The default security posture of Linux process-based containers provides a large attack surface using a shared Linux kernel. Without hardening, it may be susceptible to exploits that allow for container escape.	a. Where high-risk workloads are identified, consideration should be given to using either container runtimes that provide hypervisor-level isolation for the workload or dedicated security sandboxes	For Kubernetes clusters this would mean using alternate container runtimes like gVisor or Firecracker. With cloud managed distributions serverless container platforms like Cloud Run, Fargate or ACI would also mitigate this risk
9.2	Windows process-based containers do not provide a security barrier (per Microsoft's guidance) allowing for possible container break-out.	a. Where Windows containers are used to run application containers, Hyper-V isolation should be deployed in-line with Microsoft's security guidance.	This presents a challenge for Kubernetes clusters as, currently, Windows in Kubernetes doesn't support Hyper-V based containers
10	Patching		
10.1	Outdated container orchestration tool components can be vulnerable to exploits that allow for the compromise of the installed cluster or workloads	a. All container orchestration tools should be supported and receive regular security patches, either from the core project or back-ported by the orchestration system vendor.	In general for Kubernetes this just requires that the cluster is updated to stay within the support lifecycle for the distribution in use.

Section	Threat	Best Practice	Kubernetes Specific Information
10.2	Vulnerabilities present on container orchestration tool hosts (commonly Linux VMs) will allow for compromise of container orchestration tools and other components.	a. Host operating system of all the nodes that are part of a cluster controlled by a container orchestration tool should be patched and kept up to date. With the ability to reschedule workloads dynamically, each node can be patched one at a time, without a maintenance window.	It's important to ensure with Kubernetes cluster nodes that not only are all updates installed regularly but that updated kernels are used (this usually requires that the node(s) are rebooted)
10.3	As container orchestration tools commonly run as containers in the clusters, any container with vulnerabilities may allow compromise of container orchestration tools.	a. All container images used for applications running in the cluster should be regularly scanned for vulnerabilities, patches should be regularly applied, and the patched images redeployed to the cluster.	No specific content
11	Resource Management		
11.1	A compromised container could disrupt the operation of applications due to excessive use of shared resources	a. All workloads running via a container orchestration system should have defined resource limits to reduce the risk of "noisy neighbors" causing availability issues with workloads in the same cluster.	For Kubernetes this means that ResourceQuotas are used on all in-scope namespaces
12	Container Image Building		
12.1	Container base images downloaded from untrusted sources, or which contain unnecessary packages, increase the risk of supply chain attacks.	a. Application container images should be built from trusted, up-to-date minimal base images.	No specific content
12.2	Base images downloaded from external container image registries can introduce malware, backdoors, and vulnerabilities	a. A set of common base container images should be maintained in a container registry that is under the entity's control	No specific content

Section	Threat	Best Practice	Kubernetes Specific Information
12.3	The default position of Linux containers, which is to run as root, could increase the risk of a container breakout	a. Container images should be built to run as a standard (non-root) user.	In addition to this being specified in images, it should be enforced using Admission controllers in the Kubernetes cluster
12.4	Application secrets–i.e., cloud API credentials–embedded in container images can facilitate unauthorized access.	a. Secrets should never be included in application images. Where secrets are required during the building of an image (for example to provide credentials for accessing source code–this process should leverage container builder techniques to ensure that the secret will not be present in the final image.	No specific content
13	Registry		
13.1	Unauthorized modification of an organization’s container images could allow an attacker to place malicious software into the production container environment.	a. Access to container registries managed by the organization should be controlled. b. Rights to modify or replace images should be limited to authorized individuals.	No specific content
13.2	A lack of segregation between production and non-production container registries may result in insecure images deployed to the production environment	a. Consider using two registries, one for production or business-critical workloads and one for development/test purposes, to assist in preventing image sprawl and the opportunity for an unmaintained or vulnerable image being accidentally pulled into a production cluster	No specific content
13.3	Vulnerabilities can be present in base images, regardless of the source of the images, via misconfiguration and other methods.	a. If available, registries should regularly scan images and prevent vulnerable images from being deployed to container runtime environments.	No specific content

Section	Threat	Best Practice	Kubernetes Specific Information
13.4	Known good images can be maliciously or inadvertently substituted or modified and deployed to container runtime environments.	a. Registries should be configured to integrate with the image build processes such that only signed images from authorized build pipelines are available for deployment to container runtime environments.	No specific content
14	Version Management		
14.1	Without proper control and versioning of container orchestration configuration files, it may be possible for an attacker to make an unauthorized modification to an environment's setup	a. Version control should be used to manage all non-secret configuration files. b. Related objects should be grouped into a single file. c. Labels should be used to semantically identify objects.	No specific content
15	Configuration Management		
15.1	Container orchestration tools may be misconfigured and introduce security vulnerabilities.	a. All configurations and container images should be tested in a production-like environment prior to deployment. b. Configuration standards that address all known security vulnerabilities and are consistent with industry-accepted hardening standards and vendor security guidance should be developed for all system components, including container orchestration tools. i. Address all known security vulnerabilities. ii. Be consistent with industry-accepted system hardening standards or vendor hardening recommendations. iii. Be updated as new vulnerability issues are identified.	No specific content

Section	Threat	Best Practice	Kubernetes Specific Information
16	Segmentation		
16.1	Unless an orchestration system is specifically designed for secure multi-tenancy, a shared mixed-security environment may allow attackers to move from a low-security to a high-security environment.	a. Where practical, higher security components should be placed on dedicated clusters. Where this is not possible, care should be taken to ensure complete segregation between workloads of different security levels.	Running PCI In-scope and out-of-scope workloads on the same Kubernetes cluster presents considerable difficulties of isolation as there are a number of cluster level resources and services which cannot be easily namespaced. For example customresourcedefiition and cluster DNS services
16.2	Placing critical systems on the same nodes as general application containers may allow attackers to disrupt the security of the cluster through the use of shared resources on the container cluster node	a. Critical systems should run on dedicated nodes in any container orchestration cluster.	This can be done in Kubernetes usng node pools and admission control to ensure that workloads are only scheduled on appropriate nodes
16.3	Placing workloads with different security requirements on the same cluster nodes may allow attackers to gain unauthorized access to high security environments via breakout to the underlying node.	a. Split cluster node pools should be enforced such that a cluster user of the low-security applications cannot schedule workloads to the high-security nodes.	This can be done in Kubernetes usng node pools and admission control to ensure that workloads are only scheduled on appropriate nodes
16.4	Modification of shared cluster resources by users with access to individual applications could result in unauthorized access to sensitive shared resources.	a. Workloads and users who manage individual applications running under the orchestration system should not have the rights to modify shared cluster resources, or any resources used by another application.	In Kubernetes this means that rights must be restricted to all non-namespaced objects, additional shared services such as Cluster DNS must be protected from compromise of a single workload or node