

	<a href="#">Source</a>			41	48	85%
	Threat	Best Practice	Deckhouse approach	Deckhouse score	Best possible score	Rate
<b>1</b>	<b>Authentication</b>					
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.	According to the recommended best practice.	1	1	
1.2	Generic administrator accounts are in place for container orchestration tool management. The use of these accounts would prevent the 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.	Generic credentials are not used in the platform. Runners are authenticated through service accounts, while users through the OIDC authentication mechanism.	1	1	
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.	Certificates are only used for control-plane components authentication in the cluster and never escape control-plane nodes.	1	1	
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.	We use Kubernetes secrets or CRs to store authentication credentials. Although those resources are not encrypted by default, the user has all the mechanisms out of the box to do it on their own. We on our side are thinking about providing the secrets store out of the box: <a href="https://github.com/deckhouse/deckhouse/issues/3053">https://github.com/deckhouse/deckhouse/issues/3053</a>	0.5	1	
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.	Zero rights are provided by default. Each application or service gets the least rights required according to the cluster RBAC. All the provided rights in the cluster are stored in the Deckhouse repo, which excludes the very possibility of the rights injection. There are 6 predefined roles with the exact rules in the cluster which we suggest using.	1 1	1 1	
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).	The MFA could be connected to the cluster's OIDC, though is out of the box. Static passwords aren't used.	1	1	
<b>2</b>	<b>Authorization</b>					
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.	According to the recommended best practice.	1	1	
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.	According to the recommended best practice.	1 1	1 1	
2.3	Accounts may accumulate permissions without documented approvals.	a. Use manual and automated means to regularly audit implemented permissions.	The reconciliation loop is implemented in the platform. All the created objects in the cluster are being regularly checked for the desired state accordance.	1	1	
<b>3</b>	<b>Workload Security</b>					
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.	We have Pod Security Standards realized in the platform that defines three different policies to broadly cover the security spectrum. These policies are cumulative and range from highly-permissive to highly-restrictive.	1	1	
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, messagedigests should be used.	We plan to start deploying images using sha digests in the nearest future: <a href="https://github.com/deckhouse/deckhouse/issues/1825">https://github.com/deckhouse/deckhouse/issues/1825</a>	0	1	
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.	All container images are stored in the private repo. We build all the open-source platform components on our side from their source codes.	1	1	
<b>4</b>	<b>Network Security</b>					
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.	The recommended approach is realized based on the cilium network policies. Multitenancy out of the box will be realized in the platform in the nearest future: <a href="https://github.com/deckhouse/deckhouse/issues/1419">https://github.com/deckhouse/deckhouse/issues/1419</a>	0.5	1	
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 components and other administrative APIs should be restricted using an explicit allow-list of IP addresses.	The recommended approach is realized based on the cilium network policies. Multitenancy out of the box will be realized in the platform in the nearest future: <a href="https://github.com/deckhouse/deckhouse/issues/1419">https://github.com/deckhouse/deckhouse/issues/1419</a>	0.5	1	
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.	All the platform components interact through TLS protocol and are authorized through certificates.	1	1	
<b>5</b>	<b>PKI</b>					
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.	Certificates are only used for control-plane components authentication in the cluster and never escape control-plane nodes. Certificates are only used for control-plane components authentication in the cluster and never escape control-plane nodes.	1 1	1 1	
5.2	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.	According to the recommended best practice.	1	1	
<b>6</b>	<b>Secrets Management</b>					
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.	According to the recommended best practice.	1	1	
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.	According to the recommended best practice.	1	1	
<b>7</b>	<b>Container Orchestration Tool Auditing</b>					
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.	The platform collects the audit logs which could be stored in the external storage by log-shipper module.	1	1	
<b>8</b>	<b>Container Monitoring</b>					
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.	Log-shipper implements the centralized logging of container activity. The Loki logs storage will be part of the platform in the nearest future: <a href="https://github.com/deckhouse/deckhouse/issues/2532">https://github.com/deckhouse/deckhouse/issues/2532</a>	0.5	1	
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.	All the cluster activities are logged to the audit logs by log-shipper and Falco will be implement as the analysis tool in the nearest future: <a href="https://github.com/deckhouse/deckhouse/issues/2423">https://github.com/deckhouse/deckhouse/issues/2423</a>	0.5	1	
<b>9</b>	<b>Container Runtime Security</b>					
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 hypervisorlevel isolation for the workload or dedicated security sandboxes.	According to the recommended best practice. Open Policy Agent is integrated to check if the pod is launched with privileged or excessive rights.	1	1	

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.	Windows containers are not supported.	1		1	
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 backported by the orchestration system vendor.	We regularly check CVEs, renew base images for all the components, and patch Kubernetes to the latest minor versions.	1		1	
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.	The platform is not responsible for the security updates on the operating system level.	0		1	
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.	We check vulnerabilities by the trivy component on the daily basis. You can check reports <a href="#">here</a> .	1		1	
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.	Open Policy Agent controls resource's requests and limits for all the launching containers. Multitenancy will be realized and will allow configuring the default resources: <a href="https://github.com/deckhouse/deckhouse/issues/1419">https://github.com/deckhouse/deckhouse/issues/1419</a>	0.5		1	
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.	We build all the platform images on our own.	1		1	
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.	All platform images are stored in the private registry.	1		1	
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.	It's checked by linter on the platform level.	1		1	
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.	We provide the license tokens which allow access to the images registry. According to the recommended best practice.	1 1		1 1	
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.	We build separate images for development and production environments.	1		1	
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.	We regularly check images by the trivy component. We plan to automate it in the nearest future: <a href="https://github.com/deckhouse/deckhouse/issues/2679">https://github.com/deckhouse/deckhouse/issues/2679</a>	0.5		1	
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.	The list of all platform images is stored in the Deckhouse container, whose manifest is installed from a trusted source. Content-based hashes are used as tags. We plan to start deploying images using sha digests in the nearest future: <a href="https://github.com/deckhouse/deckhouse/issues/1825">https://github.com/deckhouse/deckhouse/issues/1825</a>	0.5		1	
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.	According to the recommended best practice. According to the recommended best practice. All the platform-deployed components are explicitly labeled.	1 1 1		1 1 1	
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.	According to the recommended best practice. We regularly check containers and binaries by Trivy. The process will be automated in the nearest future: <a href="https://github.com/deckhouse/deckhouse/issues/2679">https://github.com/deckhouse/deckhouse/issues/2679</a> Additionally, the code is scanned by linters.	1 0.5		1 1	
16 Segmentation							
16.1	Unless an orchestration system is specifically designed for secure multitenancy, 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.	It's possible to place the higher security components outside of the cluster manually and we plan to realize this option out of the box in the future. The multitenancy approach will be also realized in the platform: <a href="https://github.com/deckhouse/deckhouse/issues/1419">https://github.com/deckhouse/deckhouse/issues/1419</a>	0.5		1	
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.	We have dedicated nodes for control-plane, system needs (such as monitoring), ingress and workloads.	1		1	
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.	The workloads could be launched only on the dedicated worked nodes. Different node groups could be created and configured by taints and node selectors according to the security needs.	1		1	
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.	Out-of-the-box roles are created in accordance with the needed rights splitting, such as 'User' or 'Cluster administrator'.	1		1	