Threat matrix for Kubernetes

By Yossi Weizman

Published: 2020-04-02 · Archived: 2026-04-05 12:34:35 UTC

Updated on May 10, 2021: An updated version of the threat matrix for containers is available here.

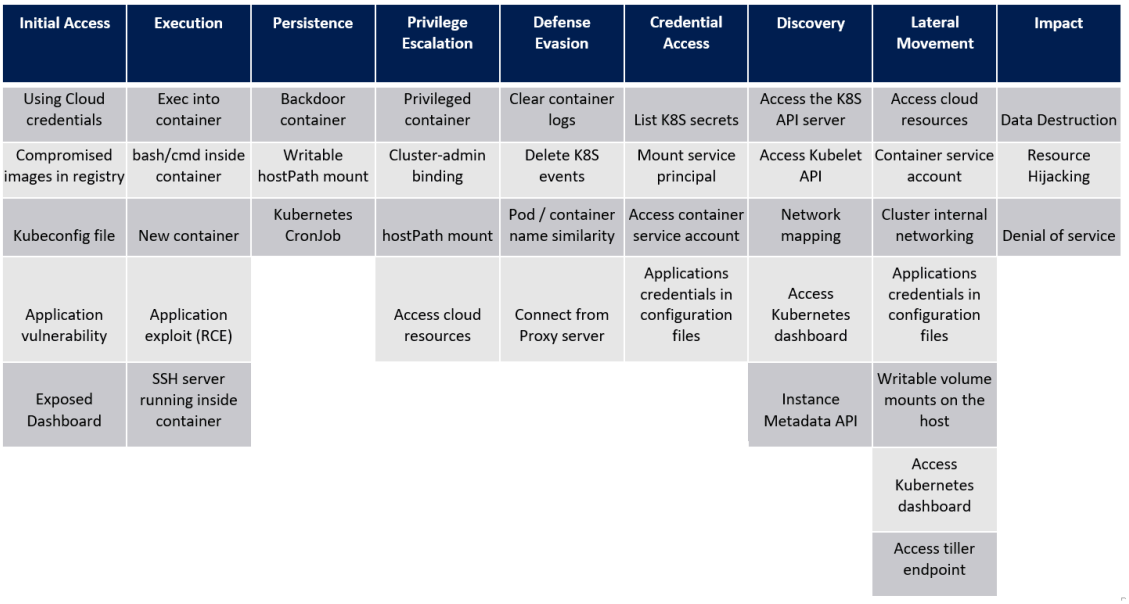
Kubernetes, the most popular container orchestration system and one of the fastest-growing projects in the history of open source, becomes a significant part of many companies' compute stack. The flexibility and scalability of containers encourage many developers to move their workloads to Kubernetes. While Kubernetes has many advantages, it also brings new security challenges that should be considered. Therefore, it is crucial to understand the various security risks that exist in containerized environments, and specifically in Kubernetes.

The MITRE ATT&CK® framework is a knowledge base of known tactics and techniques that are involved in cyberattacks. Started with coverage for Windows and Linux, the matrices of MITRE ATT&CK cover the various stages that are involved in cyberattacks (tactics) and elaborate the known methods in each one of them (techniques). Those matrices help organizations understand the attack surface in their environments and make sure they have adequate detections and mitigations to the various risks. MITRE ATT&CK framework tactics include:

- Initial access
- Execution
- Persistence
- Privilege escalation
- Defense evasion
- Credential access
- Discovery
- Lateral movement
- Impact

When we in Azure Security Center started to map the security landscape of Kubernetes, we noticed that although the attack techniques are different than those that target Linux or Windows, the tactics are actually similar. For example, a translation of the first four tactics from OS to container clusters would look like 1. "initial access to the computer" becomes "initial access to the cluster", 2. "malicious code on the computer" becomes "malicious activity on the containers", 3. "maintain access to the computer" becomes "maintain access to the cluster", and 4. "gain higher privileges on the computer" becomes "gain higher privileges in the cluster".

Therefore, we have created the first Kubernetes attack matrix: an ATT&CK-like matrix comprising the major techniques that are relevant to container orchestration security, with focus on Kubernetes.



Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Impact
Using Cloud credentials	Exec into container	Backdoor container	Privileged container	Clear container logs	List K8S secrets	Access the K8S API server	Access cloud resources	Data Destruction
Compromised images in registry	bash/cmd inside container	Writable hostPath mount	Cluster-admin binding	Delete K8S events	Mount service principal	Access Kubelet API	Container service account	Resource Hijacking
Kubeconfig file	New container	Kubernetes CronJob	hostPath mount	Pod / container name similarity	Access container service account	Network mapping	Cluster internal networking	Denial of service
Application vulnerability	Application exploit (RCE)		Access cloud resources	Connect from Proxy server	Applications credentials in configuration files	Access Kubernetes dashboard	Applications credentials in configuration files	
Exposed Dashboard	SSH server running inside container					Instance Metadata API	Writable volume mounts on the host	
							Access Kubernetes dashboard	
							Access tiller endpoint	

As can be seen, the matrix contains the 9 tactics listed above. Each one of them contains several techniques that can be used by attackers to achieve different goals. Below are the descriptions of each one of the techniques.

1. **Initial Access**

1.

The initial access tactic consists of techniques that are used for gaining access to the resource. In containerized environments, those techniques enable first access to the cluster. This access can be achieved directly via the cluster management layer or, alternatively, by gaining access to a malicious or vulnerable resource that is deployed on the cluster.

- Using cloud credentials

In cases where the Kubernetes cluster is deployed in a public cloud (e.g., AKS in Azure, GKE in GCP, or EKS in AWS), compromised cloud credential can lead to cluster takeover. Attackers who have access to the cloud account credentials can get access to the cluster's management layer.

- Compromised images in registry

Running a compromised image in a cluster can compromise the cluster. Attackers who get access to a private registry can plant their own compromised images in the registry. The latter can then be pulled by a user. In addition, users often use untrusted images from public registries (such as Docker Hub) that may be malicious.

Building images based on untrusted base images can also lead to similar results.

- Kubeconfig file

The kubeconfig file, also used by kubectl, contains details about Kubernetes clusters including their location and credentials. If the cluster is hosted as a cloud service (such as AKS or GKE), this file is downloaded to the client

via cloud commands (e.g., “az aks get-credential” for AKS or “gcloud container clusters get-credentials” for GKE).

If attackers get access to this file, for instance via a compromised client, they can use it for accessing the clusters.

- Vulnerable application

Running a public-facing vulnerable application in a cluster can enable initial access to the cluster. A container that runs an application that is vulnerable to remote code execution vulnerability (RCE) may be exploited. If service account is mounted to the container (default behavior in Kubernetes), the attacker will be able to send requests to the API server using this service account credentials.

- Exposed dashboard

The Kubernetes dashboard is a web-based user interface that enables monitoring and managing a Kubernetes cluster. By default, the dashboard exposes an internal endpoint (ClusterIP service). If the dashboard is exposed externally, it can allow unauthenticated remote management of the cluster.

2. **Execution**

The execution tactic consists of techniques that are used by attackers to run their code inside a cluster.

- Exec into container

Attackers who have permissions, can run malicious commands in containers in the cluster using exec command (“kubectl exec”). In this method, attackers can use legitimate images, such as an OS image (e.g., Ubuntu) as a backdoor container, and run their malicious code remotely by using “kubectl exec”.

- New container

Attackers may attempt to run their code in the cluster by deploying a container. Attackers who have permissions to deploy a pod **or a controller** in the cluster (such as DaemonSet \ ReplicaSet\ Deployment) can create a new resource for running their code.

- Application exploit

An application that is deployed in the cluster and is vulnerable to a remote code execution vulnerability, or a vulnerability that eventually allows code execution, enables attackers to run code in the cluster. If service account is mounted to the container (default behavior in Kubernetes), the attacker will be able to send requests to the API server using this service account credentials.

- SSH server running inside container

SSH server that is running inside a container may be used by attackers. If attackers gain valid credentials to a container, whether by brute force attempts or by other methods (such as phishing), they can use it to get remote access to the container by SSH.

3. **Persistence**

The persistence tactic consists of techniques that are used by attackers to keep access to the cluster in case their initial foothold is lost.

- Backdoor container

Attackers run their malicious code in a container in the cluster. By using the Kubernetes controllers such as DaemonSets or Deployments, attackers can ensure that a constant number of containers run in one, or all, the nodes in the cluster.

- Writable hostPath mount

hostPath volume mounts a directory or a file from the host to the container. Attackers who have permissions to create a new container in the cluster may create one with a writable hostPath volume and gain persistence on the underlying host. For example, the latter can be achieved by creating a cron job on the host.

- Kubernetes CronJob

Kubernetes Job is a controller that creates one or more pods and ensures that a specified number of them successfully terminate. Kubernetes Job can be used to run containers that perform finite tasks for batch jobs. Kubernetes CronJob is used to schedule Jobs. Attackers may use Kubernetes CronJob for scheduling execution of malicious code that would run as a container in the cluster.

4. **Privilege escalation**

The privilege escalation tactic consists of techniques that are used by attackers to get higher privileges in the environment than those they currently have. In containerized environments, this can include getting access to the node from a container, gaining higher privileges in the cluster, and even getting access to the cloud resources.

- Privileged container

A privileged container is a container that has all the capabilities of the host machine, which lifts all the limitations regular containers have. Practically, this means that privileged containers can do almost every action that can be performed directly on the host. Attackers who gain access to a privileged container, or have permissions to create a new privileged container (by using the compromised pod's service account, for example), can get access to the host's resources.

- Cluster-admin binding

Role-based access control (RBAC) is a key security feature in Kubernetes. RBAC can restrict the allowed actions of the various identities in the cluster. Cluster-admin is a built-in high privileged role in Kubernetes. Attackers who have permissions to create bindings and cluster-bindings in the cluster can create a binding to the cluster-admin ClusterRole or to other high privileges roles.

- hostPath mount

hostPath mount can be used by attackers to get access to the underlying host and thus break from the container to the host. (See "3: Writable hostPath mount" for details).

- Access cloud resources

If the Kubernetes cluster is deployed in the cloud, in some cases attackers can leverage their access to a single container in order to get access to other cloud resources outside the cluster. For example, in AKS each node contains service principal credential that is stored in **/etc/kubernetes/azure.json**. AKS uses this service principal to create and manage Azure resources that are needed for the cluster operation.

By default, the service principal has contributor permissions in the cluster's Resource Group. Attackers who get access to this service principal file (by hostPath mount, for example) can use its credentials to access or modify the cloud resources.

5. **Defense evasion**

The defense evasion tactic consists of techniques that are used by attackers to avoid detection and hide their activity.

- Clear container logs

Attackers may delete the application or OS logs on a compromised container in an attempt to prevent detection of their activity.

- Delete Kubernetes events

A Kubernetes event is a Kubernetes object that logs state changes and failures of the resources in the cluster. Example events are a container creation, an image pull, or a pod scheduling on a node.

Kubernetes events can be very useful for identifying changes that occur in the cluster. Therefore, attackers may want to delete these events (e.g., by using: "kubectl delete events–all") in an attempt to avoid detection of their activity in the cluster.

- Pod / container name similarity

Pods that are created by controllers such as Deployment or DaemonSet have random suffix in their names. Attackers can use this fact and name their backdoor pods as they were created by the existing controllers. For example, an attacker could create a malicious pod named *coredns-{random suffix}* which would look related to the CoreDNS Deployment.

Also, attackers can deploy their containers in the kube-system namespace where the administrative containers reside.

- Connect from proxy server

Attackers may use proxy servers to hide their origin IP. Specifically, attackers often use anonymous networks such as TOR for their activity. This can be used for communicating with the applications themselves or with the API server.

6. **Credential access**

The credential access tactic consists of techniques that are used by attackers to steal credentials.

In containerized environments, this includes credentials of the running application, identities, secrets stored in the cluster, or cloud credentials.

- List Kubernetes secrets

A Kubernetes secret is an object that lets users store and manage sensitive information, such as passwords and connection strings in the cluster. Secrets can be consumed by reference in the pod configuration. Attackers who have permissions to retrieve the secrets from the API server (by using the pod service account, for example) can access sensitive information that might include credentials to various services.

- Mount service principal

When the cluster is deployed in the cloud, in some cases attackers can leverage their access to a container in the cluster to gain cloud credentials. For example, in AKS each node contains service principal credential. (See “4: Access cloud resources” for more details.)

- Access container service account

Service account (SA) represents an application identity in Kubernetes. By default, an SA is mounted to every created pod in the cluster. Using the SA, containers in the pod can send requests to the Kubernetes API server. Attackers who get access to a pod can access the SA token (located in */var/run/secrets/kubernetes.io/serviceaccount/token*) and perform actions in the cluster, according to the SA permissions. If RBAC is not enabled, the SA has unlimited permissions in the cluster. If RBAC is enabled, its permissions are determined by the RoleBindings \ ClusterRoleBindings that are associated with it.

- Application credentials in configuration files

Developers store secrets in the Kubernetes configuration files, such as environment variables in the pod configuration. Such behavior is commonly seen in clusters that are monitored by Azure Security Center. Attackers who have access to those configurations, by querying the API server or by accessing those files on the developer’s endpoint, can steal the stored secrets and use them.

7. **Discovery**

The discovery tactic consists of techniques that are used by attackers to explore the environment to which they gained access. This exploration helps the attackers to perform lateral movement and gain access to additional resources.

- Access the Kubernetes API server

The Kubernetes API server is the gateway to the cluster. Actions in the cluster are performed by sending various requests to the RESTful API. The status of the cluster, which includes all the components that are deployed on it, can be retrieved by the API server. Attackers may send API requests to probe the cluster and get information about containers, secrets, and other resources in the cluster.

- Access Kubelet API

Kubelet is the Kubernetes agent that is installed on each node. Kubelet is responsible for the proper execution of pods that are assigned to the node. Kubelet exposes a read-only API service that does not require authentication (TCP port 10255). Attackers with network access to the host (for example, via running code on a compromised container) can send API requests to the Kubelet API. Specifically querying *https://[NODE IP]:10255/pods/* retrieves the running pods on the node. *https://[NODE IP]:10255/spec/* retrieves information about the node itself, such as CPU and memory consumption.

- Network mapping

Attackers may try to map the cluster network to get information on the running applications, including scanning for known vulnerabilities. By default, there is no restriction on pods communication in Kubernetes. Therefore, attackers who gain access to a single container, may use it to probe the network.

- Access Kubernetes dashboard

The Kubernetes dashboard is a web-based UI that is used for monitoring and managing the Kubernetes cluster. The dashboard allows users to perform actions in the cluster using its service account (kubernetes-dashboard) with the permissions that are determined by the binding or cluster-binding for this service account. Attackers who gain access to a container in the cluster, can use its network access to the dashboard pod. Consequently, attackers may retrieve information about the various resources in the cluster using the dashboard's identity.

- Instance Metadata API

Cloud providers provide instance metadata service for retrieving information about the virtual machine, such as network configuration, disks, and SSH public keys. This service is accessible to the VMs via a non-routable IP address that can be accessed from within the VM only. Attackers who gain access to a container, may query the metadata API service for getting information about the underlying node. For example, in Azure, the following request would retrieve all the metadata information of an instance: *http:///metadata/instance?api-version=2019-06-01*

8. **Lateral movement**

The lateral movement tactic consists of techniques that are used by attackers to move through the victim's environment. In containerized environments, this includes gaining access to various resources in the cluster from a given access to one container, gaining access to the underlying node from a container, or gaining access to the cloud environment.

- Access cloud resources

Attackers may move from a compromised container to the cloud environment. (See "4: Access cloud resources" for details).

- Container service account

Attackers who gain access to a container in the cluster may use the mounted service account token for sending requests to the API server, and gaining access to additional resources in the cluster. (See “6: Access container service account” for more details.)

- Cluster internal networking

Kubernetes networking behavior allows traffic between pods in the cluster as a default behavior. Attackers who gain access to a single container may use it for network reachability to another container in the cluster.

- Applications credentials in configuration files

Developers store secrets in the Kubernetes configuration files, for example, as environment variables in the pod configuration. Using those credentials attackers may gain access to additional resources inside and outside the cluster. (See “6: Application credentials in configuration files” for more details.)

- Writable volume mounts on the host

Attackers may attempt to gain access to the underlying host from a compromised container. (See “3: Writable hostPath mount” for more details.)

- Access Kubernetes dashboard

Attackers who have access to the Kubernetes dashboard may manage the cluster resources and also run their code on the various containers in the cluster using the built-in “exec” capability of the dashboard. (See “7: Access Kubernetes dashboard” for more details.)

- Access tiller endpoint

Helm is a popular package manager for Kubernetes maintained by CNCF. Tiller is the server-side component of Helm up to version 2.

Tiller exposes internal gRPC endpoint in the cluster, listens to port 44134. By default, this endpoint does not require authentication. Attackers may run code on any container that is accessible to the tiller’s service and perform actions in the cluster, using the tiller’s service account, which often has high privileges.

9. **Impact**

The Impact tactic consists of techniques that are used by attackers to destroy, abuse, or disrupt the normal behavior of the environment.

- Data destruction

Attackers may attempt to destroy data and resources in the cluster. This includes deleting deployments, configurations, storage, and compute resources.

- Resource hijacking

Attackers may abuse a compromised resource for running tasks. A common abuse is to use compromised resources for running digital currency mining. Attackers who have access to a container in the cluster or have permissions to create new containers may use them for such activity.

- Denial of service

Attackers may attempt to perform a denial of service attack, which makes the service unavailable to the legitimate users. In container clusters, this include attempts to block the availability of the containers themselves, the underlying nodes, or the API server.

Understanding the attack surface of containerized environments is the first step of building security solutions for these environments. The matrix that was presented above can help organizations identify the current gaps in their defenses' coverage against the different threats that target Kubernetes. Azure Security Center can help you protect your containers environment. Learn more about Azure Security Center's support for container security.
