Adopting a DevSecOps Approach for Modern Apps

Recommended security practices from code to customer

Are you on a path to modernize the way you build, deliver and manage applications and services? If so, you also need to modernize the way you think about and practice security, governance and compliance.

Here is a short list of recommended security practices, roughly ordered to align to the stages of the container lifecycle—from code to customer.

You can download our **comprehensive whitepaper** for a better understanding of DevSecOps practices over the lifecycle of a container that every team should consider when modernizing.

CODE

MAKE DEVELOPING SECURE, CLOUD NATIVE SOFTWARE EASIER

Many programming frameworks make adopting recommended security practices and patterns easier for developers, enabling them to create secure applications by default. These frameworks also can help security teams implement low-level policies for how those functions are run without disrupting the application or its development.

CI

BUILD MODULAR CONTAINER IMAGES WITH LAYERS THAT CAN BE INDIVIDUALLY UPDATED

No need to rebuild containers from scratch, placing more burden on testing and validation practices. Containers should be continuously updated and redeployed with only the layers that change being rebuilt. Then you should have verifiable proof of provenance backed by auditable container metadata and automated reporting for security and compliance teams.

BUILD SECURE, CUSTOM CONTAINERS

The more unnecessary code you have in your container, the higher the likelihood that there is also an unpatched security flaw. Instead, you can standardize the code used in the base OS, as well as application dependencies. Metadata provides documentation of image provenance, as well as a means for policy enforcement and monitoring.

SECURE USE OF THIRD-PARTY CONTAINERS

Today, developers can choose to source software from a variety of places, but it's difficult to know whether this containerized software adheres to your organization's policies. You need a private container registry for managing approved container images and base OS images—including associated metadata for all images. And access and deployment policies for that private registry should be rigorously controlled.

ADOPT CONTROL POLICIES FOR IMAGES IN YOUR ENVIRONMENT

Organizations that allow teams to pull and run applications from any registry into production are vulnerable to threats, whether from unpatched CVEs or from malware embedded within the image layers. With a private registry, operations teams can validate images, set usage policies, and keep images refreshed and updated so that only compliant images can run in production.

CD

SECURE THE CONTAINER RUNTIME PLATFORMS

Securing clusters and applying ubiquitous and uniform policies across your Kubernetes estate is a major challenge for teams. Organizations should implement a zero-trust, role-based access control policy for accessing each runtime platform, with credentials stored off platform. Explicit control policies for each platform allow only containers from trusted sources to be run on production clusters.

CONTINUOUSLY UPDATE THE INFRASTRUCTURE LAYER

Keeping your Kubernetes clusters updated and aligned to open source can be burdensome for teams. You can use a centralized system to notify when clusters across the organization require upgrades. This limits the security risk from unpatched clusters. Then use a uniform API, such as Cluster API, to streamline the process of upgrading, as well as to execute more complex events like backup and recovery.

USE A CENTRALIZED CONTROL PLANE TO MANAGE CONTAINER RUNTIME SPRAWL

Various flavors of Kubernetes clusters can proliferate across an organization and make it difficult to manage all instances holistically. A centralized management system for clusters can enable an organization to provide uniform, self-service access to new Kubernetes clusters while applying consistent, policy-based management—with the flexibility for teams to set policies for individual clusters, as necessary.

SECURE INTER-CONTAINER COMMUNICATIONS

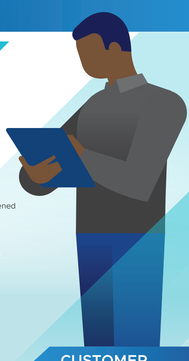
Inter-container communications can be vulnerable to outside threats and should be secured and monitored. A service mesh can provide authorization and encryption features needed to secure communications and protect data in transit. Securing communications is a lot like setting up access permissions: Containers should only be allowed to talk to the containers they need to do their job—nothing more, and nothing less.

OBSERVE EVERYTHING

Monitoring applications has significantly changed, with applications now split across many microservices and clusters. When an organization grows to hundreds or thousands of containers, it is no longer possible to monitor individual components using legacy systems. Through programmatic observability of the microservice application logs, attackers scanning or trying to exploit applications can be identified easily.

GET A TRUSTED DEVSECOPS PARTNER

To achieve the level of oversight and automation needed to build and maintain a secure container lifecycle, organizations need to have the right tools and capabilities. The **VMware Tanzu™ Advanced** edition is designed to help organizations create a secure container lifecycle from start to finish—from automated container builds to applications running safely in production on hardened Kubernetes clusters to the management of policies across clusters and clouds.



CUSTOMER

Get the security outcomes your organization needs:

- Reduce time to remediate security vulnerabilities
- Improve visibility and control of container contents and lifecycle
- Reduce toil for security, compliance, and DevOps teams

LEARN MORE ABOUT SECURING THE CONTAINER LIFECYCLE.

DOWNLOAD THE WHITEPAPER

vmware®