10 Things You Must Do to Secure Containers Now

Get Started

vmware®

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

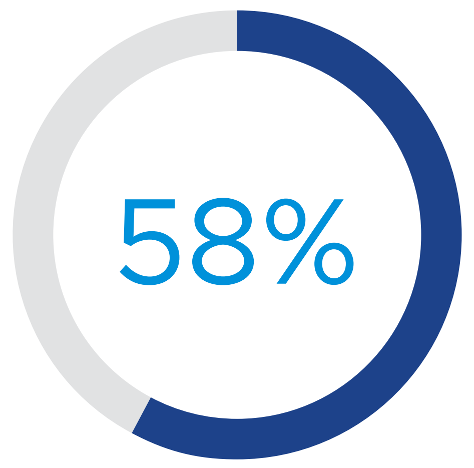
Conclusion and Next Steps >

Containers and other cloud-native technology drive massive organizational and operational changes. Organizations of all sizes use this technology to deliver modern applications, be more competitive and drive more revenue through faster innovation and better customer experiences. Containers give organizations clear advantages: they go from code to customer faster and more regularly.

Today, Kubernetes is the most widely used platform to manage containerized applications at scale. According to the State of Kubernetes Report 2022[1], over half of organizations surveyed plan to expand their Kubernetes environments within the next year. While there are countless advantages gained from container and Kubernetes adoption, like any new technology they bring new security risks that have not gone unnoticed by cybercriminals.

The rise of containerized applications has made an appealing case for threat actors to pursue these dynamic attack surfaces. Due to its popularity, cybercriminals are now creating attacks specifically targeting Kubernetes vulnerabilities. The increased speed of development and the ephemeral nature of containers means Developers have less time to implement security protocols, leaving points of entry open for attackers. Additionally, the use of public image registries has become commonplace, and many Developers use third-party sourced components. Recognizing an opportunity, attackers are targeting these open-source projects and inserting malicious code early in the software supply chain to maximize their reach.

The attack surface for containerized applications is growing rapidly relative to virtualized applications, yet DevOps and Security teams face complexity when running containers at scale in areas like policy management, visibility and networking. Making matters worse, traditional security tools and manual processes are outdated and ineffective for securing these environments. To properly assess and manage security risk, DevSecOps teams need holistic visibility and modern solutions made for cloud native environments. Additionally, organizations face the cultural shift in adopting a DevSecOps mindset to bolster collaboration. This e-book will outline ten best practices for developing a container security strategy, threat profiles for common attacks and pro tips for strengthening security posture.



of the financial industry's top CISOs and security leaders witnessed an increase in application attacks in 2021.[2]

1. VMware | State of Kubernetes Report 2022
2. VMware | Modern Bank Heist Report

Introduction >
1 Align Security Controls with the Application Lifecycle >
2 Adopt a DevSecOps Mindset >
3 Secure Every Layer of Infrastructure >
4 Reduce Risk of Open-Source Software >
5 Minimize Risk when Using Third-Party Image Registries >
6 Reduce the Attack Surface >
7 Implement Runtime Security >
8 Secure Your Network >
9 Adopt Trusted Compliance and Security Frameworks >
10 Stay Informed >
Conclusion and Next Steps >

Align Security Controls with the Application Lifecycle

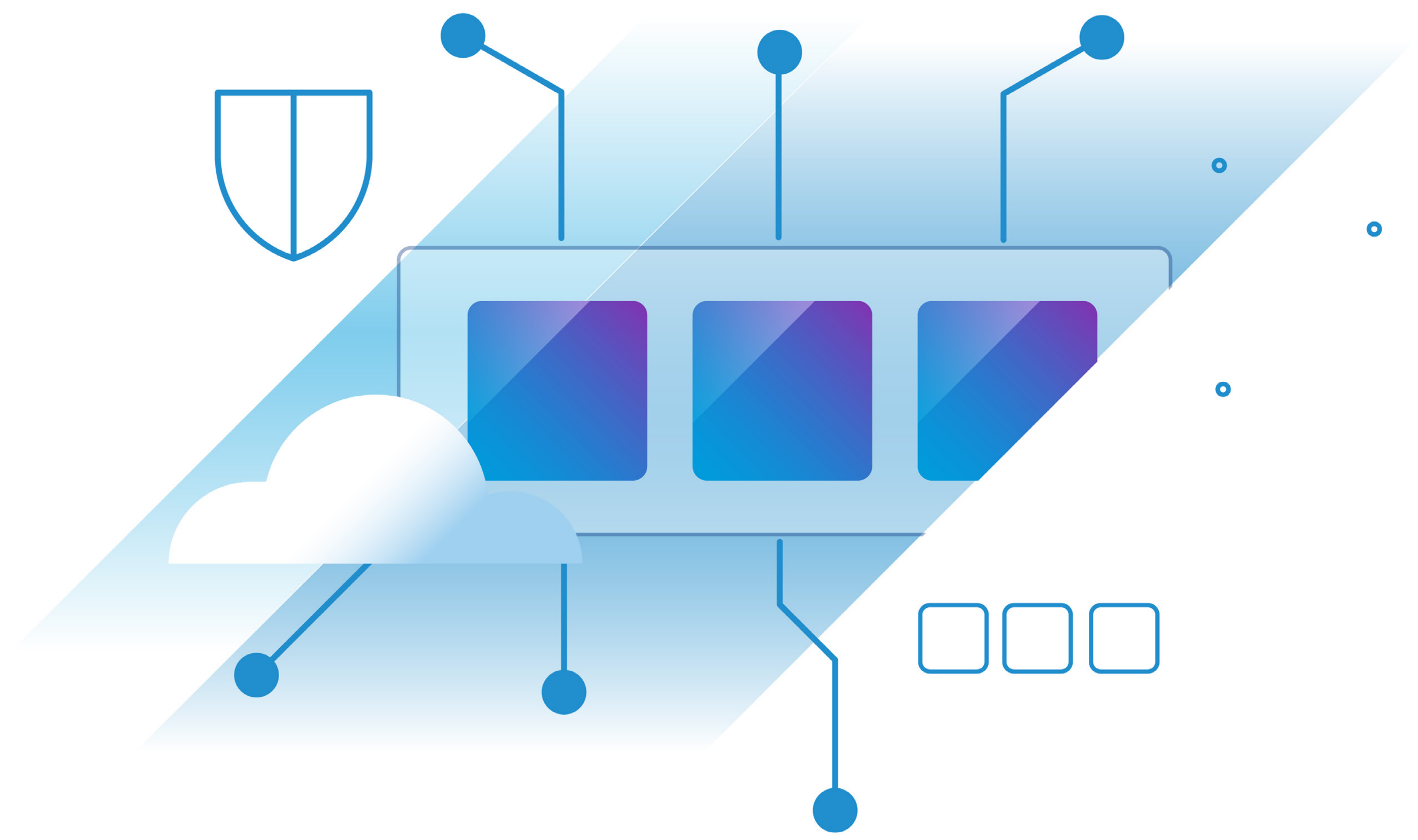
Each step in the application lifecycle requires different techniques and tools to ensure the application is built in a secure way. Even before Development teams begin coding and building an application, the first step must be critical planning. DevOps teams should determine those involved in the process, their assigned roles, what tools to use, and what security requirements must be met.

Comprehensive security that will not impede developers requires tightly orchestrated and automated solutions that should:

1. Provide streamlined access management and onboarding
2. Ensure compliance with security policy and align with industry standards
3. Support business continuity
4. Emphasize transparency and open communication between teams
5. Remove complexity

Pro Tip: Shifting Left

The term "shift-left" means security starts at the development phase of an application. Through shifting left, Developers are coding securely at the beginning of the application lifecycle, ensuring security is a shared responsibility. "Shifting security left" early in the development cycle allows Security and DevOps teams to seamlessly automate container security from build to production while supporting developer agility, modernization, and operational efficiencies.



vmware®

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

Conclusion and Next Steps >

Build

To code securely, Development teams need application building blocks that are fortified and trusted. Developers must work with the Security team to patch and test reported vulnerabilities, while closely monitoring third-party dependencies. Before moving to the next phase, container images must be scanned for vulnerabilities and misconfigurations to meet hardening and compliance requirements. Fixing vulnerabilities in the build phase can save developers a lot of time compared to later phases.

Deploy

At the deployment phase, Kubernetes configuration errors could unintentionally allow the container to run with escalated privileges or host access, exposing it to attacks. DevOps teams must validate the workload manifest complies with security policies based on Kubernetes security best practices and industry benchmarks such as the Center for Internet Security (CIS) and the Security Technical Implementation Guides (STIG) before deployment. If the file is correctly configured according to the security policy, it can move into production. Automated, continuous scanning throughout the CI/CD pipeline enables DevOps teams to deploy applications faster and more securely.

Operate

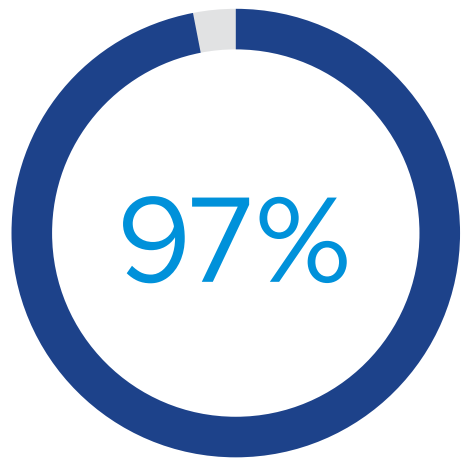
Effective container security in the Operate phase must address runtime image scanning as well as threat detection and response to understand the overall security posture and manage risk. Runtime scanners ensure the environment is free of risky configurations and vulnerabilities, and the DevSecOps team can verify all workloads are monitored, or denied for security risks. Security teams require visibility to correlate between hardening, runtime and vulnerability scanning data to enforce compliance and security governance.

Introduction
1 Align Security Controls with the Application Lifecycle
2 Adopt a DevSecOps Mindset
3 Secure Every Layer of Infrastructure
4 Reduce Risk of Open-Source Software
5 Minimize Risk when Using Third-Party Image Registries
6 Reduce the Attack Surface
7 Implement Runtime Security
8 Secure Your Network
9 Adopt Trusted Compliance and Security Frameworks
10 Stay Informed
Conclusion and Next Steps

Adopt a DevSecOps Mindset

DevSecOps is a methodology enabling Development, Security, and IT Operations teams to deliver secure software at an agile pace by embedding security throughout the entire software development lifecycle. This strategy helps drive uniformity and collaboration between these three teams.

What happens if there is no collaboration? In the same way a car would not be built without going through a safety inspection, the same can be said about the development process of a new application. It is crucial for the application to go through an "inspection" during every step of the process. The tools an organization chooses needs to support this model in order to be more effective and reduce any potential friction between teams.



of organizations have concerns about Kubernetes security.

1. Source: VMware | State of Kubernetes Report 2022

Introduction
1 Align Security Controls with the Application Lifecycle
2 Adopt a DevSecOps Mindset
3 Secure Every Layer of Infrastructure
4 Reduce Risk of Open-Source Software
5 Minimize Risk when Using Third-Party Image Registries
6 Reduce the Attack Surface
7 Implement Runtime Security
8 Secure Your Network
9 Adopt Trusted Compliance and Security Frameworks
10 Stay Informed
Conclusion and Next Steps

Benefits of a DevSecOps Approach

When security is embedded through every step of the application lifecycle, everyone benefits.

Developers

With a DevSecOps model, developers can become increasingly agile and push code into production faster. Instead of starting anew if an application does not meet security and compliance requirements, they save time with this "secure by design" model. Catching security-related flaws early in the development process is also much less costly to the organization.

Security Teams

Security teams can benefit from the increased transparency this methodology creates. They have visibility into what developers are coding, where they are sourcing code, and any potential vulnerabilities. They can enforce compliance requirements and remediate potential risks faster.

Operations Teams

Operations teams can benefit through scalability and automated workflows.

DevSecOps is vital when it comes to building cloud-native applications as these applications are inherently more scalable than their legacy counterparts. Traditional security practices—especially those involving human intervention—often don't address the complexities of cloud-native applications. As a result, cloud-first companies quickly adopt the use of containers running as microservices to reap significant value and scale company growth.

Pro Tip

Implementing a DevSecOps approach should not be a hurried process. If the end goal is to automate and become more agile, the beginning is a vital time to figure out what tools and processes are needed:

1. Create a team with representatives from each area (Development, Security, IT Operations).
2. Start small. Identify a small set of projects that can benefit the most from embedding security into the lifecycle. Look for the most dynamic teams with a well-established process.
3. Start with open-source tools. The cloud-native open-source community is flourishing with active projects on security, with different tools to try until you better understand your needs. Once requirements are established, an enterprise grade commercial tool should be implemented.
4. Integrate into the broader business once ready, and regularly work to improve functions.
5. Stay up to date on security best practices, industry standards and critical vulnerabilities.

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

Conclusion and Next Steps >

Secure Every Layer of Infrastructure

Attackers are actively exploiting the additional entry points modern applications provide. A strong security posture must go beyond the containers and be incorporated throughout each layer of infrastructure. Because new attacks can emerge at any given time, it is critical the process of layered security is done continuously. A layered security approach will ensure an application is built secure and stays secure throughout its entire lifecycle.

Pro Tip
Incorporating security into each layer the application is running on will help establish a strong security posture. Look for security solutions that reduce risk across multiple layers of your infrastructure.

OS layer

Vulnerabilities in the operating system (OS) layer occur in any environment where the host OS lives for the container. Host OS risks have the largest attack surface, and these risks are some of the most critical to protect. Some examples of risks specific to this layer include:

- Kernel vulnerabilities
- Vulnerable packages
- Open ports allowing remote access (ssh, rsh, ftp, etc)
- Permissive privilege escalation (sudo configuration)
- Permissive access to secrets (encryption keys, cloud access tokens, etc)

Container layer

The container layer is usually the first and the most common place to mitigate risk, and for good reason. Container images are reused and distributed across development groups, so a single flawed or compromised container image may impact many workloads. In fact, many container security solutions only solve for these specific risks. Types of risks that occur in the container layer include:

- Images with critical vulnerabilities
- Containers running with the privileged flag
- Unrestricted communications between containers
- Containers running rogue or malicious processes
- Containers not properly isolated from the host

Introduction >
1 Align Security Controls with the Application Lifecycle >
2 Adopt a DevSecOps Mindset >
3 Secure Every Layer of Infrastructure >
4 Reduce Risk of Open-Source Software >
5 Minimize Risk when Using Third-Party Image Registries >
6 Reduce the Attack Surface >
7 Implement Runtime Security >
8 Secure Your Network >
9 Adopt Trusted Compliance and Security Frameworks >
10 Stay Informed >
Conclusion and Next Steps >

Kubernetes layer

Kubernetes is the most widely used container orchestration tool, allowing users to define the desired end state of their applications via logical constructs such as deployments, replica sets, configuration maps, services, and more. Kubernetes users are exposed to attacks on the Kubernetes server which can have major ramifications, and application configuration risks including:

- Remote access to workloads from misconfiguration of services such as load balancers and node ports
- Secrets exposure and embedding secrets in workload specs
- Credential theft
- Secrets sitting in the hands of users who no longer need them
- Resource permissions within the cluster
- Access to cloud resources that allow entry to databases and other cloud service accounts
- Privilege escalation as a result of misconfigured Role-Based Access Control (RBAC)

Cloud layer

Reducing misconfigurations, monitoring malicious activity, and preventing unauthorized access are foundational activities necessary to ensure security and compliance of applications and data in the cloud. As criminals become more sophisticated in their abilities to exploit cloud misconfiguration vulnerabilities, security teams need a smarter approach to prevent security breaches. Risks in the cloud layer include:

- API vulnerabilities
- Misconfigurations
- Insufficient identify and access management controls

Threat Profile: Exploiting Kubernetes RBAC Misconfigurations

Role-Based Access Control (RBAC) is the ability to set permission for users that limits their ability to certain networks based on their role. In Kubernetes, the RBAC module is used to help specify which workloads and clusters can access specific data. Because Kubernetes manages a wide variety of objects each with its own risks, it must only allow necessary functions and restrict unnecessary access. This strategy is commonly referred to as a least-privilege approach, and typically included as part of a zero-trust implementation. For example, you may think providing “read only” access to a user seems harmless, but it can give read access to secrets, including service account token for service accounts with higher privilege.

Introduction
1 Align Security Controls with the Application Lifecycle
2 Adopt a DevSecOps Mindset
3 Secure Every Layer of Infrastructure
4 Reduce Risk of Open-Source Software
5 Minimize Risk when Using Third-Party Image Registries
6 Reduce the Attack Surface
7 Implement Runtime Security
8 Secure Your Network
9 Adopt Trusted Compliance and Security Frameworks
10 Stay Informed
Conclusion and Next Steps

Reduce Risk of Open-Source Software

Open-source software (OSS) has transformed the world of software development and is a major contributor to the rise of containers and microservices. Without OSS, organizations operated in a "closed box" model. Developers can now leverage vast functionality already developed elsewhere in other organizations with similar needs. Open-source offers Developers the power to narrow their focus to the areas that bring value to their organization. However, the ways OSS is consumed (source code, OS and language-specific packages, container images) introduce massive risk to an organization.

In fact, cybercriminals now target specific open-source projects to inject malicious code into the software supply chain. They can take a different approach to each, such as exploiting vulnerabilities to inject backdoors or malware. Malware can be some of the most common attacks in open-source, but at the same time some of the hardest to detect.

Organizations can no longer rely on the open-source community alone to validate shared software. The volume and potential risk of security flaws—and now targeted attack insertion—require a systematic approach to reducing the risk associated with the broad adoption of OSS.

Pro Tip

Reduce open-source software risk through source scanning, container image scanning, and utilizing only trusted registries and repositories.

Threat Profile: Common Exploit

The December 2021 Log4j vulnerability is a prime example of exploited code in an OSS. Log4j allows Developers to log user activity and is a universally used tool by Apache Software. Attackers were able to execute code remotely on a server and thus steal data or take control of the system. Organizations around the globe were affected, and in one study, over 90,000 public-facing servers running on OSS contained a vulnerable version of Log4j.[1]

1. DARKReading | Log4j Attack Surface Remains Massive

Introduction

1 Align Security Controls with the Application Lifecycle

2 Adopt a DevSecOps Mindset

3 Secure Every Layer of Infrastructure

4 Reduce Risk of Open-Source Software

5 Minimize Risk when Using Third-Party Image Registries

6 Reduce the Attack Surface

7 Implement Runtime Security

8 Secure Your Network

9 Adopt Trusted Compliance and Security Frameworks

10 Stay Informed

Conclusion and Next Steps

Minimize Risk when Using Third-Party Image Registries

Third-party image registries help Developers scale quickly. They save time through automated deployments and allow for collaboration across teams and applications. When building an image, it can either be run on the system it was built on or uploaded to a registry and downloaded onto another computer. Certain registries are public, allowing anyone to pull images, while others are private, only accessible to certain people or machines.

Public image registries are typically used by smaller teams, requiring less resources and budget to set up and maintain. While public image registries can be extremely useful, they are inherently less secure. When using a public image registry, take extra security precautions to reduce the risk of an application.

Cybercriminals love to insert malicious code into image registries. If a developer pulls from a public registry and uses that malicious code in an application, the application is now susceptible to attack. Automation helps developers push updates more frequently, but automated deployments of this code in the continuous integration/continuous deployment (CI/CD) pipeline can further cultivate misconfigurations or malicious images in production. Additionally, a once secure image can become vulnerable after it is added to the pipeline.

Pro Tip

Use third-party registries to ease resource strains, but reduce risk with continuous scanning for vulnerabilities and misconfigurations throughout the CI/CD pipeline, before deployment and at runtime. Follow security best practices, adopt industry standards, and ensure container images meet organizational security requirements at each stage of development.

Threat Profile: Delivering Poisoned Container Images

One of the easiest ways for developers to consume OSS is to pull an image that includes the application they need, whether it is networking, monitoring, data storage or other service. Such images contain many different files, and an attacker can easily hide malware an image and publish it. Their hope is a developer will pull it into their clusters, bypassing all perimeter defenses, where the malware can run and act as a beachhead for an attack.

vmware®

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

Conclusion and Next Steps >

Reduce the Attack Surface with Vulnerability Management and Hardening

Image scanning is the process of scanning container images for vulnerabilities throughout the application lifecycle. Scanners help identify any potential threats to your environment so security teams can take the necessary actions to mitigate those risks. Most scanners use the Common Vulnerabilities & Exposures (CVE) database to identify these potential threats.

Pro Tip:
Image scanning needs to take place at each stage of the application lifecycle, even after it is deployed. The ability to restrict image registries is also an important tool for preventing developers from using registries that are not approved.

An image scanning tool should:

- **Give visibility** into the container images in your environment
- **Provide insight** into identified vulnerabilities and available fixes
- **Create exceptions** at the image level from inside the image scan report
- **Prevent container images** with vulnerabilities from progressing through the CI/CD pipeline
- **Ensure container images** used in any running workload are up to date and approved

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

Conclusion and Next Steps >

Scanning in the CI/CD pipeline and at runtime

Image scanning needs to take place everywhere, for every image. Effective image scanning must occur at each phase of the development lifecycle — from Build through Operate. The ability to carry vulnerability scanning through the runtime layer gives Security and DevOps teams visibility into vulnerabilities in images that were not previously scanned, and images deployed from any third-party registries. Giving the user control to customize their scan means Security teams can introduce automated, custom policies and be notified when those policies have been violated. Automation is a key component to helping organizations achieve continuous compliance.

A strong need for visibility

DevSecOps teams need visibility and a deep understanding of the workloads running in their environments in order to properly secure them. They need an image scanning tool that helps them understand the current level of risk and protect against potential threats and weaknesses in their environment. Ideally, this tool should allow them to

- Review vulnerable images in the CI/CD pipeline
- Review images running in production
- Prioritize vulnerabilities by severity
- Ensure all images are scanned
- Restrict registries and repositories allowed in production
- Get visibility into connectivity and configuration of applications installed in Kubernetes clusters

Threat Profile: Trojan Malware

Container image malware is one of the most common types of exploits related to the use of third-party image registries. Specifically, Trojan Malware is a type of malware disguised as useful code and can easily go unnoticed. To prevent this type of attack, using a trusted image scanner is critical.

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

Conclusion and Next Steps >

Implement Runtime Security

Containers and cloud-native infrastructure and applications are constantly evolving, which leads to a wide variety of vulnerabilities. In order to detect attacks, focusing on known patterns can only be part of the solution. Identifying what constitutes normal application behavior, and detecting suspicious deviations from that behavior, is key for reducing the rate of false positives and detecting exploits of zero-day vulnerabilities.

What makes threat detection in cloud-native environments different than traditional detection and response is that there is no longer an affinity between the application and server. Any server can run multiple applications, any application can run on multiple servers, and that relationship is constantly changing.

A cloud-native detection and response solution continuously collects comprehensive activity data of cloud services, container orchestrators, servers and applications, giving you the information needed to:

- Proactively hunt threats
- Uncover suspicious behavior
- Disrupt attacks in progress
- Repair damage quickly
- Manage vulnerabilities
- Address gaps in defenses

Threat Profile: Cryptomining

Why are Kubernetes clusters a favorite target for cryptominers? Many organizations do not protect Kubernetes or monitor Kubernetes security events, leaving them vulnerable. When a container runs on a modern server with a certain number of vCPUs, the miner can use all vCPUs if no quota is applied on the compromised container. It is important to remember if a Kubernetes cluster is hosted on an elastic platform, there is no limit for the CPU, and your cloud bill will be grossly inflated.

Currently, XMRig is the most popular cryptominer on Linux. It is both open-source, available in Ubuntu packages, and as a ready to run container on DockerHub. Regardless of the primary attack vector, injection, vulnerability in SQL, log4j, or any vulnerability in an application used in a container, the attacker needs to inject and run the malicious application, in our case XMRig, inside a container of the victim Kubernetes cluster. The default network policy for Kubernetes is “Allow All,” meaning by default any container can connect to any local/Internet IP.

Pro Tip

Scanning open ports is the standard tactic for attackers to find the next step and move laterally through an environment, and detecting such behavior is critical to stopping a breach. Look for security solutions that can detect and alert on port scanning activity.

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

Conclusion and Next Steps >

Runtime security addresses all aspects of running containers in production. Security and DevOps teams need a single source of truth to connect data and help detect and address runtime anomalies. A consolidated view of these events enables better investigation and correlation of different types of events for different types of objects.

There are two major categories of runtime threats:

- Attacks that use your data or infrastructure to run malicious activities
- Attacks that use your infrastructure to mine cryptocurrency

Threat detection is especially critical for stopping lateral attacks. In addition, it is important that security tools eliminate noise and alert on real and active events without affecting the application and overall user experience. Implementing the right level of runtime security ensures your applications can remain safe from threats – including those outside the physical control of security teams.

How can you mitigate runtime attacks?

Before the attack:

- Identify container images that have known software vulnerabilities
- Harden Kubernetes configuration: use least privilege security principles
- Build a baseline of container behaviors: network/ file access
- Reduce the attack surface: eliminate complexity, minimize the amount of code
- Limit the blast radius: leverage micro segmentation and encryption

During the attack:

- Block/detect network IP/URLs and malicious files on a blacklist
- Block/detect deviations from the baselines
- Block/detect privilege escalation
- Block/detect suspicious behaviors: use of shells, download of binaries/packages, change in configuration files

vmware®

Introduction
1 Align Security Controls with the Application Lifecycle
2 Adopt a DevSecOps Mindset
3 Secure Every Layer of Infrastructure
4 Reduce Risk of Open-Source Software
5 Minimize Risk when Using Third-Party Image Registries
6 Reduce the Attack Surface
7 Implement Runtime Security
8 Secure Your Network
9 Adopt Trusted Compliance and Security Frameworks
10 Stay Informed
Conclusion and Next Steps

Secure Your Network

In today's hyper-connected world, applications are becoming more virtualized and distributed across many locations, some outside the physical control of security teams. With the number of attacks on companies rising, protecting network traffic and infrastructure is critical.

Ingress and Egress Security

Traffic can flow in and around your network in various ways and security teams need to be on high alert for where that traffic is coming from. Ingress data refers to unsolicited traffic coming from outside an organization's network that gets transferred in. Service-to-service connections, or egress data, can be used to leak information from the cluster layer. It is important to gain visibility into these connections and have controls on both ends – ingress controls will help minimize hostile incoming traffic, while egress controls prevent insiders from sharing data to unauthorized groups.

API Security

API security risks are a concern not to take lightly, as APIs are such a common and universal method of passing information. New API breaches and vulnerabilities are constantly being discovered, and traditional approaches to API security are too slow to keep up with the evolving tactics by malicious actors. According to VMware's 2022 Global Incident Response Threat Report, 23% of all attacks seen by respondents in the past 12 months compromised API security[1].

Threat Profile: Exploiting Rogue and Open APIs

Rogue or shadow APIs are APIs that exist outside of an organization's official security and operational maintenance processes, and are particularly dangerous as they can be exploited to steal data or gain access to an organization's systems. Many teams have open APIs that are publicly accessible to customers or partners, but an unsecured open API means cybercriminals can integrate a virus themselves directly into its code. DevSecOps teams must track all APIs that are in use in order to identify and mitigate risk.

1. VMware | Service Mesh for Dummies, VMware 2nd Special Edition

Introduction

1 Align Security Controls with the Application Lifecycle

2 Adopt a DevSecOps Mindset

3 Secure Every Layer of Infrastructure

4 Reduce Risk of Open-Source Software

5 Minimize Risk when Using Third-Party Image Registries

6 Reduce the Attack Surface

7 Implement Runtime Security

8 Secure Your Network

9 Adopt Trusted Compliance and Security Frameworks

10 Stay Informed

Conclusion and Next Steps

All API traffic should be encrypted, authenticated, and routed through an API gateway. Extra layers of protection can bring stronger security measures for safer API usage such as enforcing schema validations, anomaly detection, and payload inspection. Strong telemetry and visibility into an application's stack can lead to these informed decisions.

Service Mesh

A service mesh is a modern connectivity and security runtime platform that takes care of service-to-service communication and security, observability, and resiliency. It allows development teams to focus on building business logic rather than dealing with the connectivity and security requirements of an application. They do, however, need to follow security and compliance guidelines to mitigate insider threats and reduce the risk of a data breach. This can be achieved by ensuring all communications between applications are encrypted and mutually authenticated. Developers can do that in middleware by setting up a mutual Transport Layer Security (TLS) connection each time a microservice connects to another, or a client connects to a microservice.

Access Control

A multilayered security architecture that implements network security will have elements of access control and threat control. If a bad actor gains access to a network, they can monitor traffic and map the entire infrastructure. From there, they can launch an attack or insert malware. Access control restricts the movement of bad actors throughout the network. Unfortunately, that is not enough, and problems can still arise, which brings the need for threat control. Threat control prevents the actions of bad actors from doing damage within the network.

Pro Tip

Protect your network traffic and infrastructure to enable your developers to focus on building an application rather than the connectivity requirements. Use the security frameworks in the following section to ensure your protection is compliant with the standards.

1. VMware | Global Incident Response Threat Report: Weathering the Storm

Introduction
1 Align Security Controls with the Application Lifecycle
2 Adopt a DevSecOps Mindset
3 Secure Every Layer of Infrastructure
4 Reduce Risk of Open-Source Software
5 Minimize Risk when Using Third-Party Image Registries
6 Reduce the Attack Surface
7 Implement Runtime Security
8 Secure Your Network
9 Adopt Trusted Compliance and Security Frameworks
10 Stay Informed
Conclusion and Next Steps

Adopt Trusted Compliance and Security Frameworks

CIS Benchmarks

The Center for Internet Security (CIS) is an independent group of cybersecurity experts who aim to make the internet safe. CIS Benchmarks are a commonly used third-party approach to securing your environment beyond standard configuration. Organizations of all industries, sizes, and various environments use CIS Benchmarking for implementing their security standards. CIS outlines multiple documents that include guidelines around network, cloud, mobile devices and OS security, and more. Two key documents to get familiar with when it comes to container security are:

- CIS Kubernetes Benchmarks
- CIS Docker Benchmarks

NIST Framework

Another way for organizations to improve their security posture is through the National Institute of Standards and Technology (NIST) Framework. The NIST Framework allows users to measure their current cybersecurity maturity against industry standards and visualize a desired security state. The NIST Application Container Security Guide gives a detailed breakdown of how to secure containers from build to operate, including explanations of potential security concerns associated with using containers. Some helpful tips from the guide include:

- Using container-specific host OSs instead of general-purpose ones to reduce attack surfaces
- Segmenting containers by purpose and threat posture on a single host OS kernel to allow for additional defense
- Using container-aware runtime defense tools

Pro Tip:
Consider your unique environment and use this guide to decide which framework is best suited for your needs. These widely accepted industry benchmarks and frameworks will help your organization be better prepared for common attacks targeting containers and Kubernetes.

Introduction
1 Align Security Controls with the Application Lifecycle
2 Adopt a DevSecOps Mindset
3 Secure Every Layer of Infrastructure
4 Reduce Risk of Open-Source Software
5 Minimize Risk when Using Third-Party Image Registries
6 Reduce the Attack Surface
7 Implement Runtime Security
8 Secure Your Network
9 Adopt Trusted Compliance and Security Frameworks
10 Stay Informed
Conclusion and Next Steps

Stay Informed to Prepare for Common Critical Security Risks

Each year the **Open Web Application Security Project®** (OWASP) compiles a list for application developers for what is widely accepted as the top ten most critical security risks for web applications in that given year. Let's dive into the most recent list to learn more about each of these risks, why they are so common, and how to prevent them.

1. **Broken Access Control**
   Failure to enforce access control can lead to unauthorized information disclosure, destruction of all data, or a user performing a business function outside their limits. This can be prevented via deny by default or implementing access control mechanisms once and re-using them throughout the application.

2. **Cryptographic Failures**
   Failures related to cryptography (or lack thereof) can often lead to the exposure of sensitive data. This can be prevented through multiple methods, including encrypting all sensitive data at rest, and not storing sensitive data unnecessarily.

3. **Injection**
   Applications are susceptible to attack when hostile data is used, or user-supplied data is not validated by the application. Preventing the risk of injection calls for keeping data independent from commands and queries.

4. **Insecure Design**
   Risks can be related to design and architectural flaws that are missing or have ineffective controls. Establishing a secure development lifecycle to aid with evaluating and designing security and privacy-related controls can help prevent this.

5. **Security Misconfiguration**
   Systems are at higher risk if there is no concrete, replicable security configuration process. Create a secure process on a minimal platform to make it quick and simple to deploy another environment that is properly locked down.

6. **Vulnerable and Outdated Components**
   Vulnerable and outdated components can occur if teams are not keeping up to date on the state of their software. By initiating a continuous plan for monitoring, triaging, and applying updates for the lifetime of an application, security teams can be confident knowing there are no vulnerable components in their environment.

Introduction >
1 Align Security Controls with the Application Lifecycle >
2 Adopt a DevSecOps Mindset >
3 Secure Every Layer of Infrastructure >
4 Reduce Risk of Open-Source Software >
5 Minimize Risk when Using Third-Party Image Registries >
6 Reduce the Attack Surface >
7 Implement Runtime Security >
8 Secure Your Network >
9 Adopt Trusted Compliance and Security Frameworks >
10 Stay Informed >
Conclusion and Next Steps >

7. Identification and Authentication Failures

Confirming a user's identity and authentication is critical to protecting against authentication-related attacks. A weak password or ineffective credential recovery can poke holes for automated attacks. Implementing multi-factor authentication and having strict password policies can prevent these types of intrusions.

8. Software and Data Integrity Failures

A software and data integrity failure is associated with code and infrastructure that neglects to protect against integrity violations or comes from untrusted sources. Prevent failure by verifying all sources and having a review process for code and configuration changes.

9. Security Logging and Monitoring Failures

If a system is not logging and monitoring, breaches cannot be detected. Logs of applications should be monitored for suspicious activity, and any auditable events should be logged in a manner that log management solutions can easily consume.

10. Server-Side Request Forgery

Server-Side Request Forgery (SSRF) issues happen whenever a web application is fetching a remote resource without authenticating the user-given URL. This can be prevented at an application layer by verifying all client-given input data and disabling HTTP redirections.

Pro Tip

Threats to your containerized applications will happen. Be prepared for how to mitigate and respond to them by understanding how they work, and take the first step towards changing the software development culture within your organization into one that produces more secure code.

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

Conclusion and Next Steps >

Conclusion and Next Steps

Containers and cloud-native technology will allow organizations to move from code to customer faster than ever before, deliver new revenue-generating features more often, and hyper-focus on the customer experience to be more competitive. The rapid adoption of containers for modern application delivery will continue to grow, and so will the attack surface. This security guide was created to help DevSecOps teams take advantage of all the benefits containers have to offer, while mitigating the massive amount of risk that comes with using them.

Keep these ten best practices in mind as you build out your container security strategy:

1. **Align security controls with the application lifecycle**
   Security should be integrated throughout the application lifecycle, from build to run.
2. **Establish a DevSecOps approach**
   By embracing a devsecops mindset, organizations can leverage growing advances in applications without putting themselves or customers at risk.
3. **Secure every layer**
   Security must go beyond the container and be integrated throughout each layer of infrastructure. Layered security helps keep applications secure throughout their lifecycle.
4. **Reduce risk of open-source software**
   Open-source software has enabled developers to share solutions to common problems, but they are tempting targets for attackers. Prevent any risk from using this software so developer teams can focus on the areas that will bring value to your organization.
5. **Minimize risk when using third-party image registries**
   Image registries allow developers to scale quickly by automating deployments and encouraging collaboration. However, be aware of what code may be laying in a public registry.

Introduction >

1 Align Security Controls with the Application Lifecycle >

2 Adopt a DevSecOps Mindset >

3 Secure Every Layer of Infrastructure >

4 Reduce Risk of Open-Source Software >

5 Minimize Risk when Using Third-Party Image Registries >

6 Reduce the Attack Surface >

7 Implement Runtime Security >

8 Secure Your Network >

9 Adopt Trusted Compliance and Security Frameworks >

10 Stay Informed >

Conclusion and Next Steps >

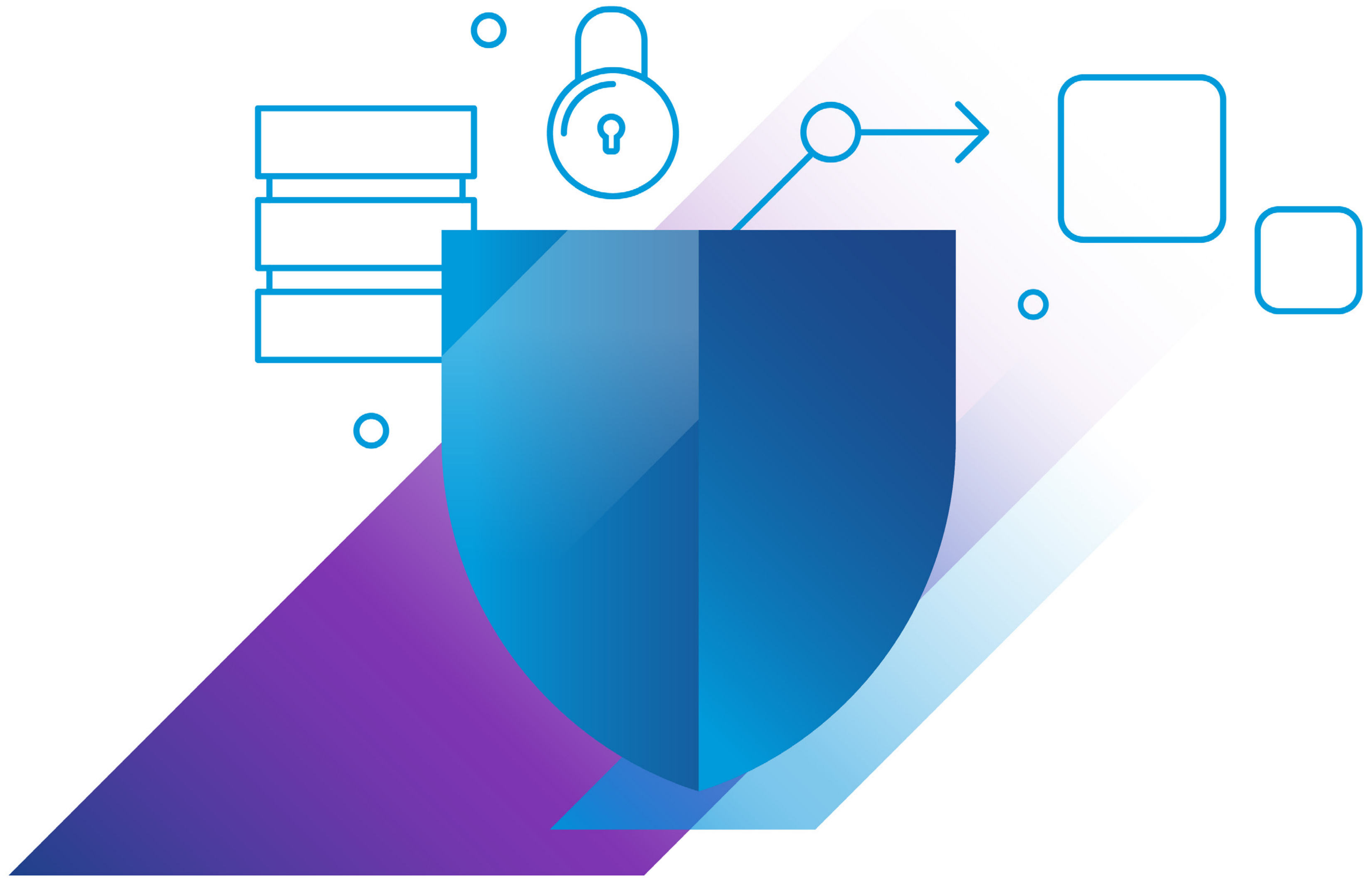
6. **Reduce the attack surface with vulnerability management and hardening**
   It is important to have a complete understanding of risks in your environment to help reduce the attack surface. Image scanning in every phase for every image provides the level of visibility needed.

7. **Secure containers at runtime**
   A consolidated view of runtime anomalies will enable strong investigation and correlation. Continuous scanning at runtime will ensure no new misconfigurations or vulnerabilities are introduced.

8. **Secure your network**
   As applications become more virtualized and distributed, it is important to protect network traffic. Understanding API security and utilizing service mesh can help verify all pieces are talking and making informed decisions.

9. **Align with industry standards and compliance frameworks**
   Improve your security posture and build a more mature security program by following security frameworks such as CIS benchmarks and the NIST framework.

10. **Prepare for critical security risks**
    Containers may be an enticing target, but you can be prepared to protect your business by following OWASP's list of the top ten most critical security risks to applications.



Check out these additional resources to learn more about securing containers:

Container Security Pathfinder

VMware Carbon Black Container Technical Overview

VMware Carbon Black Container Test Drive

Secure Modern Applications Industry Guide

Secure Modern Applications Solution Guide

Start Bridging the Dev-Sec-Ops Divide

Join us online: