WHITEPAPER

Bitdefender®

Security

A practical guide to effective container security

www.bitdefender.com

Introduction

As with mainframes, server blades, virtualization and cloud architectures before, containers offer a further resource and configuration abstraction that organizations can leverage to improve efficiency and benefit from lower total cost of ownership for business workloads.

However, any new technology requires new ways of ensuring manageability and, arguably more importantly, security of th e new way of working.

Yet, it is common for organizations to adopt new technologies without fully assessing their ability to secure the new ecosystem. This can expose them to a high risk to revenue at worst and affecting the efficiency of delivery teams and DevOps agility at best.

As containers have evolved, the decisions you need to make regarding how you host, run, and secure them have evolved. Our container experts have developed, tested, and refined this guidance and best practices for security in your environment.

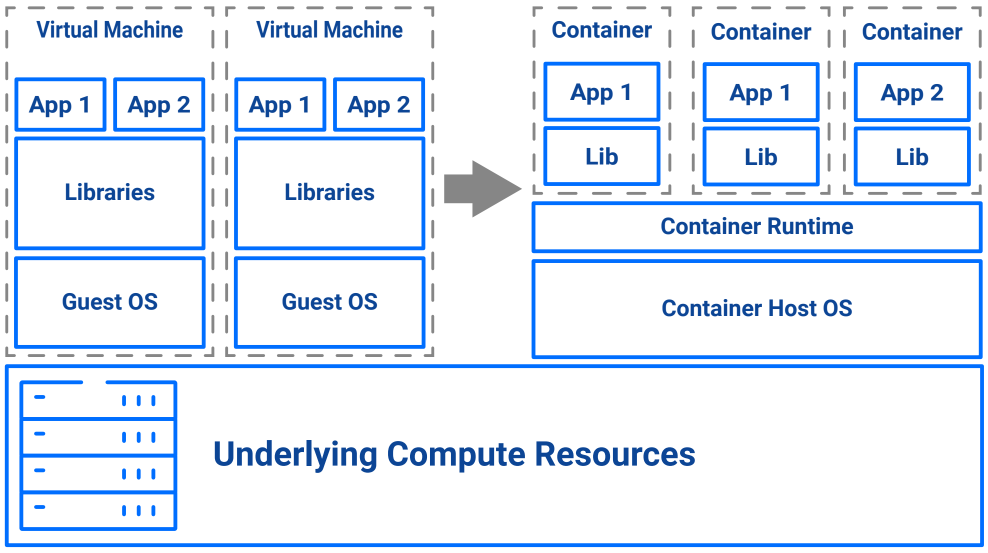
What are containers?

Often compared with virtual machines, containers provide similar benefits but in a way that impacts how they are managed and secured.

With virtual machines, a guest operating system, such as Linux or Windows, runs on a host operating system with 'virtual' access to the **underlying hardware** through an abstraction layer (the hypervisor).

With containers, a collection of libraries and other dependencies run in a packaged environment with 'virtual' access to the underlying operating system through an abstraction layer (the container runtime).

Virtualization and containers are primarily seen as complementary technologies rather than competing ones.



Instead of virtualizing the hardware stack, containers virtualize at the operating system level, with multiple containers running directly on top of the kernel.

This means that containers are far more lightweight, start faster, make it easier to scale applications and use a fraction of the resources than running an entire operating system on top of another operating system.

Securing containers 101

This intrinsic connectivity between the container and the underlying operating system causes the perception that they are less secure.

There is some truth in this – for example, if there is a vulnerability in the host, it could provide a way for an attacker to access the containers running on that host. While that is also true with virtualization present a much smaller attack surface to the guest operating system.

There are also security benefits to be gained.

The container model generally results in a more distributed application architecture, meaning that insecure containers can be replaced faster and with greater ease.

Also, reducing unnecessary modules and libraries at the container level can help reduce the attack surface.

However, the shared underlying components in the container host become an even more critical part of that attack surface.

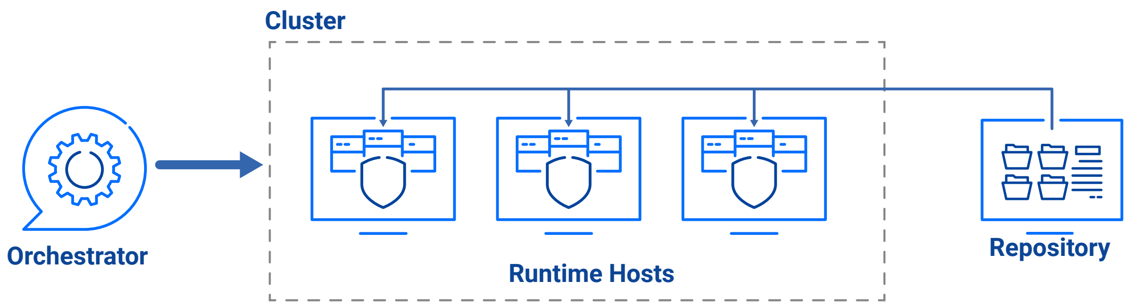
No matter what container architecture you choose, securing the application, and application components, running inside the container during runtime is still critical.

Container Architectures

The choices for hosting and managing containers vary significantly depending on the desired level of control and preferred underlying architectures.

The container ecosystem is made up of a set of building blocks, including

- **The repository**: A system which stores and manages the container images and code.
- **The container host and container runtime**: Servers that run the containers themselves and supporting components.
- **The orchestrator**: A platform that manages many containers across multiple container hosts.



There are many options for each of these, and services are available from all of the major cloud providers to orchestrate and run containers and manage images.
When we consider the security of the ecosystem, each part presents its own risks and considerations.

Securing each part of the ecosystem

Key mistakes to avoid

Starting with a mission-critical application
If you're just starting out with containers, then choose an appropriate use case to start with. For example, a back-end batch processing task or a less important customer-facing application. Stay away from revenue-generating systems.

Focusing too much on the containers themselves
While securing each container and its image is critical, don't ignore the rest of the container ecosystem. The orchestration platform, cloud environment, and container host all present an inviting target for bad actors.

Focusing too much on build-time/pre-runtime security
A robust vulnerability and configuration management practice for container images is of utmost importance, but it is not enough. A layered security plan – or defense in-depth strategy – that includes runtime threat protection, detection, and response is a must in today's threat landscape.

Ignoring automation as a fundamental requirement
Automation is critical when we consider the need to rapidly update containers when risks are discovered and allow a fast response to security incidents.

Giving containers unnecessary privileges or knowledge.
Follow a "least privilege" and "least knowledge" principle. In other words, don't give containers any more access than they need to resources, including secrets like encryption keys or other authentication details.

Failing to properly vet an image
When including an image from a public repository or building containers from a new image, be sure to fully understand its current state and its provenance. Work to identify the source of each component and its history.

Container Orchestration & Repositories

Key Risks

Container Hijacking: Server and container runtime vulnerabilities or misconfigurations present opportunities for attackers to run their own containers.

Insecure Default Configurations: Across the container ecosystem systems are configured for interoperability by default, but this can present a security risk when you place these systems on a public network

Orchestrators are an application and system like any other; they are prone to unknown or zero-day vulnerabilities.

Remote access can be compromised by an attacker to control the container ecosystem or access other sensitive data through the orchestrator.

Rogue containers can be deployed by the attacker to:

- Access other containers
- Exfiltrate data
- Consume resources through cryptojacking
- Run botnets

You should

- ☑ Tightly restrict access to the orchestrator, use an identity management provider, and do not provide public internet access if possible.
- ☑ Put controls in place to restrict what image repositories can be used by the orchestration platform
- ☑ Verify default configurations when setting up the orchestrator for the first time and after each update
- ☑ Patch and update the orchestration platform whenever a new patch is available
- ☑ If hosting the orchestration or image repository platform yourself, ensure monitoring of the hosts is in place

Tip: Many container platforms now include a signing infrastructure allowing administrators to sign container images to prevent untrusted containers from being deployed. However, remember that it is not necessarily the case that a trusted, signed container is secure to run. Vulnerabilities may be discovered in some of the software in the container after it has been signed.

Container Hosts

Key Risks

Traditional server vulnerabilities: The container host is still a server and vulnerable to the usual exploitation

Container Hijacking: Server and container runtime vulnerabilities or misconfigurations present opportunities for attackers to run their own containers.

The container host provides many underlying services to the containers themselves and also presents an attack vector itself.

Each container running on a host shares a common underlying kernel. Containers are essentially isolated from one another, which is beneficial from a security point of view. However, if the host is compromised, all containers running on it are at risk. Similarly, if a container uses a vulnerable library, it could be exploited to gain access to the underlying host.

Vulnerabilities can be present in the container runtime, as well as in the host operating system, that can allow attackers to compromise the host. There are increasing methods of operation where bad actors use clean, vanilla images and build the attack during runtime, bypassing authentication or orchestration mechanisms to access the operating system o redeploy containers directly onto the host.

You should

- ☑ Always use the most up to date version of the container runtime
- ☑ Avoid sharing host namespaces with containers.
- ☑ Ensure that there are no unnecessary ports exposed.
- ☑ Do not run ssh services inside a container.
- ☑ Avoid mounting to sensitive host directories.
- ☑ Run your containers as a non-root user.

Tip: Most Linux distributions are unnecessarily feature-heavy if their intended use is simply a host to run containers. For that reason, several Linux distributions have been designed specifically for running containers. For example

- Container Linux (formerly CoreOS)
- RancherOS
- Photon OS
- Project Atomic Host
- Ubuntu Core
- Amazon Bottlerocket

Containers & Images

Key Risks

Traditional Application & System Exploits: Anything you place inside a container that can be accessed can present a threat vector to attackers

Container Escape to host: Kernel vulnerabilities that allow arbitrary code execution may be exploited to escape a container.

Containers and images present risks to the security of your data as with any other system. When building containers, we must be as confident as possible that the base image used is as secure as possible.

As discussed previously, containers run a subset of the operating system libraries. The more tools and libraries you install into your image, the higher the risk. Only include the minimum of libraries in your image; remove anything superfluous to application dependencies.

For example: If your team needs to connect to your containers using SSH for maintenance, this creates a security risk. You should design a way to maintain containers without needing to access them directly.

If you do not want to start your image from a blank sheet, you must assess whether you trust the provider of the base image. Remember, anyone can publish an image on Docker Hub or other public repositories. At best, you could be inheriting vulnerabilities or, at worst, downloading an intentionally compromised image containing a back-door.

If you deploy to cloud environments, you can consider using “immutable deployments”. Immutable means that a container won’t be modified during its life. If you must update the application or apply a patch, you build a new image and redeploy it.

If you are ensuring images are fully patched at all times and the application architecture allows for rolling deployments, immutability can make deployments far safer.

However, we cannot eliminate all security risks in this way. New vulnerabilities are discovered every day in third-party software and, applications we build or deploy are equally likely to contain vulnerabilities in the code or their dependencies.

in the first half of 2021 over 1700 new vulnerabilities were recorded in the National Vulnerability Database every month, an average of 57 per day

The closest we can come through even the most perfect build-time security practices is to know that at a specific point in time, to the best of our knowledge, we are *probably* secure.

This means we also need to ensure that we have robust runtime protection in place. This must include monitoring the processes and activity within each container as well as the container host.

The ability to stop an attack in progress is of utmost importance but few organizations are effectively able to stop an attack or zero-day exploit before it happens.

Implement measures to stop attacks in progress and prevent zero day exploits and look to automated patching and management to provide another layer of runtime security.

You should

- ☑ Scan images for vulnerabilities prior to build.
- ☑ Keep container images up-to-date
- ☑ Avoid using privileged containers. Use fine-grained privileges instead.
- ☑ Update runtime containers as soon as possible after a vulnerability or misconfiguration is patched in an image
- ☑ Save Troubleshooting Data Separately from Containers
- ☑ Do not “bake secrets” into the image
- ☑ Deploy runtime security monitoring

And finally..

A lot of the differentiated work involved with creating a solid security strategy for your container environment versus other virtualized assets comes down to setting permissions, limiting resources, and monitoring systems. Whether your organization is just starting with containers or is already using them, we hope this guide will help you to make your solution as secure as it can be.

Bitdefender security experts are on-hand to provide guidance and advice for your on-premises, cloud, and hybrid workloads, whatever technology you’re using to support your business.

Bitdefender GravityZone Security for Containers

Container and cloud-native workload protection, detection, and response

Bitdefender GravityZone Security for Containers is a platform-agnostic, high-performance security solution for Containers and Linux OSs that combines server extended EDR with advanced Linux exploit detection and attack forensics.

It features a comprehensive kernel-independent layer stack built for Linux and containers and enables organizations to extend automation and visibility across cloud-native workloads, VMs, containers, private and public clouds, and physical endpoints.

Learn More

Bitdefender®

Built for Resilience

Bitdefender®

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

Bitdefender-Whitepaper-Containers-NoLocalization-DEX1281-en_EN
06/15/21July 26, 2021 7:34 pm07/26/21