DOCKER SECURITY: 14 BEST PRACTICES FOR SECURING DOCKER CONTAINERS

Containerization of applications involves packaging application code in a virtual container with its dependencies—the required libraries, frameworks, and configuration files. This approach aids portability and operates consistently across various computing environments and infrastructure, without losing efficiency.

One particularly popular container platform is Docker. Organizations use Docker for developing applications that are:

- Efficiently optimized
- Highly scalable
- Portable
- Agile

Through its lightweight run-time environments, Docker containers share underlying operating systems to host applications that support a DevOps environment. Being a critical element of the Cloud-Native framework, Docker brings numerous benefits to your software development lifecycle (SDLC). But those benefits aren't without risk. You're likely to face complexities, particularly when it comes to securing the Docker framework.

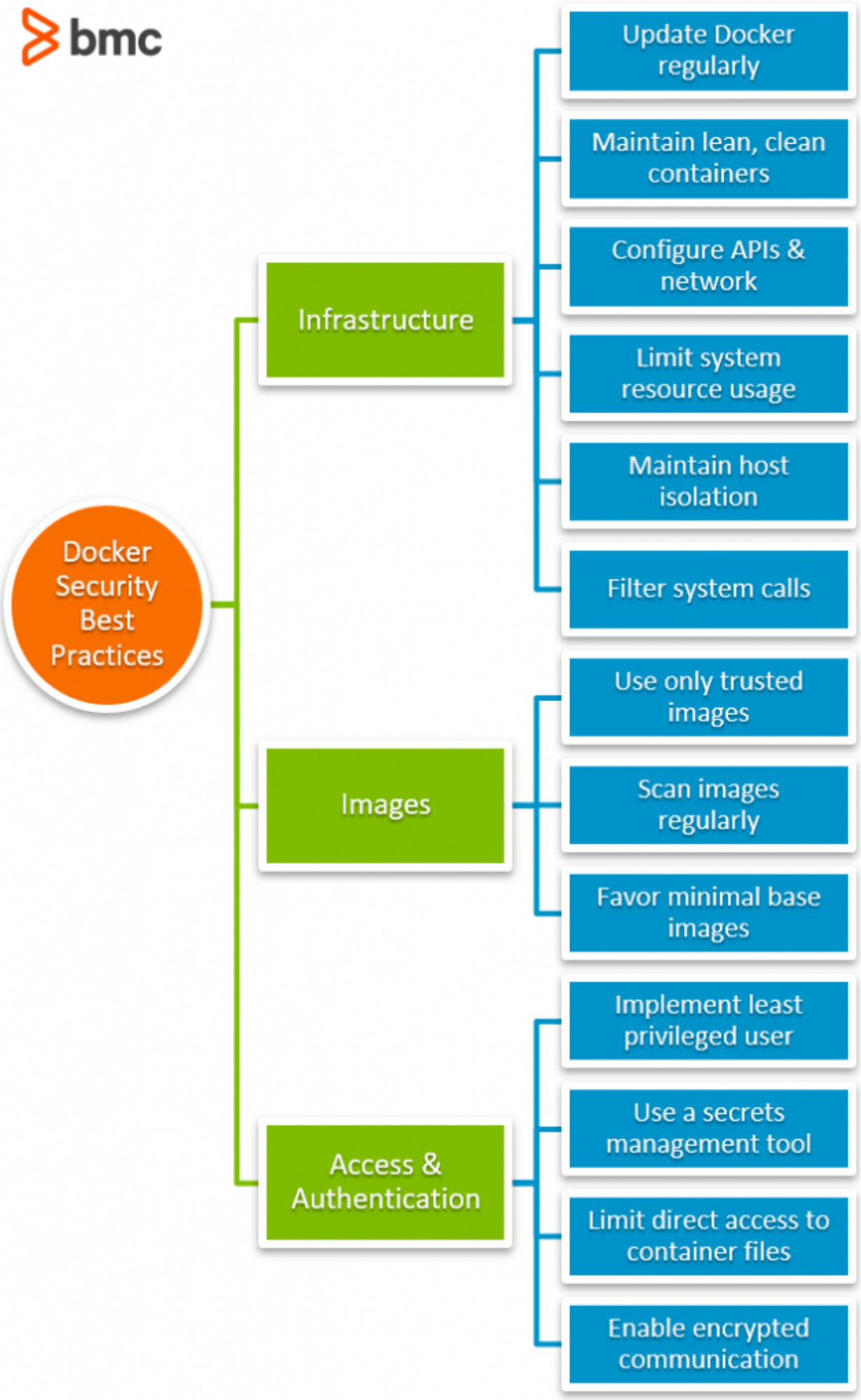
By default, Docker containers are secure. However, it is imperative that you know possible vulnerabilities in order to adopt an approach that safeguards against potential security risks.

So, in this article, we'll look at the best practices for securing a Docker-based architecture across three key areas:

- Infrastructure
- Images
- Access and authentication

Let's get started.

(



This is part of

our Docker Guide. Use the right-hand menu to navigate.)

Securing Docker infrastructure

Containers are virtualized units that can host applications. To do so, containers hold:

- Code binaries
- Configuration files
- Related dependencies

Since containers form the foundation of a cloud-native setup, securing them from potential attack vectors is a critical activity throughout the container lifecycle. A holistic approach to securing such a framework is to protect not only the Docker container but also its underlying infrastructure.

Let's break down the best approach to securing infrastructure and see how it works.

Update your Docker version regularly

First things first: Ensure that your Docker version is up to date. Obsolete versions are susceptible to security attacks. New version releases often contain patches and bug fixes that address vulnerabilities of older versions.

The same holds true for the host environment: ensure that supporting applications are up-to-date and free of known bugs or security loopholes.

Maintain lean & clean containers

An extended container environment expands the attack surface and is comparatively more prone to security breaches than lean setups. To avoid this, configure your containers to contain only the necessary components that keep them operating as you intend:

- Software packages
- Libraries
- Configuration files

Further, routinely check host instances for unused containers and base images and discard those that aren't in use.

Configure APIs & network

Docker Engine uses HTTP APIs to communicate across a network. Poorly configured APIs carry security flaws that hackers can exploit.

To avoid this, protect your containers by securely configuring the API that restricts them from being publicly exposed. One approach is to enforce encrypted communication by enabling certificate-based authentication.

(Get more details on securing Docker APIs.)

Limit usage of system resources

Set a limit on the proportion of infrastructure resources that each container can use. These infrastructure resources include:

- CPU
- Memory
- Network bandwidth

Docker uses Control Groups that limits the allocation and distribution of resources among the different processes. This approach prevents a compromised container from consuming excessive resources that could disrupt service delivery in the event of a security breach.

Maintain host isolation

Run containers with different security requirements on separate hosts.

Maintaining the isolation of containers through different namespaces serves to protect critical data from a full-blown attack. This approach also prevents *noisy neighbors* from consuming excessive resources on pool-based isolation to impact services of other containers.

Restrict container capabilities

By default, Docker containers can maintain and acquire additional privileges that may or may not be necessary to run its core services.

As a best practice, you should limit a container's permissions to only what is required to run its applications. To do so, use the command to drop all privileges of the Docker container:

```
$ docker run --cap-drop ALL
```

Following this, add specific privileges to the container with the **--cap-add** flag. This approach restricts Docker containers from obtaining unnecessary privileges that get exploited during security breaches.

Filter system calls

Apply system call filters that allow you to choose which calls can be made by containers to the Linux kernel.

This approach enables a *secure computing mode*, thereby reducing possible exposure points to avoid security mishaps—particularly to avert exploitation of Kernel vulnerabilities.

Securing Docker images

Now, let's move to security best practices beyond the infrastructure.

Docker images are templates of executable code that are used to create containers and host applications. A Docker image consists of runtime libraries and the root file system—making the image one of the most critical fundamentals of a Docker container.

Here are some best practices to follow when it comes to securing Docker images.

Use trusted image

Get Docker base images *only from trusted sources* that are up-to-date and properly configured.

Additionally, ensure Docker images are correctly signed by enabling the **Docker Content Trust** feature to filter out unsecured questionable sources.

Scan images regularly

It is crucial to maintain a robust security profile of Docker Images and routinely scan them for vulnerabilities. Do this in addition to the initial scan before downloading an image to ensure it is safe to use.

With regular image scans, you can also minimize exposure by:

- Auditing critical files and directories
- Keeping them updated with the latest security patches

Favor minimal base images

Avoid using larger *generic* Docker Images over smaller ones to minimize security vulnerabilities. This offers two valuable outcomes:

- Reduces the attack surface
- Gets rid of default configurations that are more susceptible to hacks

Access & Authentication Management

The final category for Docker Security involves access and authentication.

Securing Docker Daemon through Access Control is often known as applying the first layer of security. Without securing Docker Daemon, everything is always vulnerable:

- The underlying operations
- Applications
- Business functions

Implement least privileged user

By default, processes within Docker containers have **root** privileges that grant them administrative access to both the container and the host. This opens up containers and the underlying host to security vulnerabilities that hackers might exploit.

To avoid these vulnerabilities, set up a least-privileged user that grants only the necessary privileges to run containers. Alternatively, restrict run-time configurations that prohibit the use of a privileged user.

Use a secrets management tool

Never store secrets in a Dockerfile that may allow a user with access to the Dockerfile to misplace, misuse, or compromise an entire framework's security.

Standard best practice is to safely encrypt key secrets in third-party tools, such as the Hashicorp Vault. You can use this same approach for other container secrets beyond access credentials.

Limit direct access to container files

Transient containers require consistent upgrades and bug fixes. As a result, such container files are exposed each time a user accesses them.

As a best practice, maintain container logs outside the container. This drastically reduces consistent direct usage of container files. It also enables your team to troubleshoot issues without accessing logs within a container directory.

Enable encrypted communication

Limit Docker Daemon's access to only a handful of key users. Additionally, limit direct access to container files by enforcing SSH-only access for general users.

Use TLS Certificates for encrypting host-level communication. It's also essential to disable unused ports and keep default ports exposed only for internal use.

Securing Docker secures your IT environment

Security within an IT landscape is a critical mission that you should never overlook.

To secure a cloud-native framework, the first step always is to factor in the vulnerabilities of your framework's key elements. As a result, organizations should maintain a robust security profile that centers around containers and their underlying infrastructure.

Though approaches to implementing end-to-end security may differ, the goal is always to factor in vulnerable points and adopt best practices that mitigate risks.

RELATED READING

- BMC DevOps Blog
- BMC Multi-Cloud Blog
- Docker Management Tips
- Kubernetes vs Docker: A Quick Comparison
- How To Run MongoDB as a Docker Container