5 Container Security Best Practices You Must know

Containers are no longer the next big thing in the industry. They are mainstream now. This means they should be treated seriously and not like a novelty.

What are the main areas of concern when you want to add a new tool to your stack? For most of us stability, security, and observability are the things that matter the most. This article will focus on security.

You might recall that containers are isolated both from themselves and from the underlying Operating System. That fact may fool you into thinking that they are thus secure by design. This is only partly true. While containers themselves make it easier to design secure systems, the applications they host can make it easier or harder for an attacker to gain unauthorized access. What's more, even if containers solve some security issues, they can introduce new ones. We will show you how to make sure you use containers in the best way possible.

Securing Images

Containers are created from container images. Making sure to limit the possibility of malicious activity on those images is the best way to start.

Images contain a subset of an operating system together with an application that would be running in a container. Each tool and each library you pull into your image is a potential threat. Therefore the best way to avoid such threats is to only include your application in a container image. Ideally, a statically compiled binary containing all of the dependencies. If that's not feasible, try to include as little as possible. Trim down everything your application won't be using like the "vi" editor or the "sed" and "awk" binaries. This way you reduce the attack surface.

And if you want to base your image on anything other than "scratch" you have to decide whether you trust the provider of the base image. Anyone can publish an image on Docker Hub. Even if you think it might be a good idea to save yourself some time and build on top of somebody else's work, there are better ways than simply pulling an image. You can try to recreate the steps by inspecting the `Dockerfile` and only include those parts you feel are trustworthy.

Another thing to remember is the separation of privileges. The ancient rule of not running your application with root privileges is well known and widely accepted. Yet, when it comes to containers, people tend to forget about it.

Don't make it easy for an attacker by serving him a root account on a silver platter. Instead, run the application from a dedicated user account inside the container. It’s as easy as adding two lines to your `Dockerfile`!

Securing Registries

You’ve built an image that you secured in the best way possible. Now is the time to store it in a registry. It may look simple enough, but security considerations also apply here.

There are several ties between registries and security. The first is access control. This is the most basic form of selecting who can and who cannot publish new images. But you can add another layer of security by signing your images. Signed images can be traced back to the person who signed them, which means it is harder to substitute them for ones that have been compromised. You can learn more about image signing by reading the Docker Content Trust document. Docker also provides an open-source tool called Notary that helps with signing and verifying images.

If you store your images in a registry you should scan them regularly for vulnerabilities as well. But even after performing these vulnerability checks when building the image, some vulnerabilities may only be discovered in the future. So the image that was once free of concerns suddenly becomes a threat. Performing scans and audits on all the images you store will help you discover such issues when they arise.

Whether you host your own registry or use a managed one, try to assess how secure you think it is. How many people have access? How could one gain unauthorized access? In this regard, containers are not different from more traditional applications. They are simply artifacts that should be protected.

Securing Deployment

There are multiple deployment strategies and each can present threats that are absent from the others. There’s one thing in common regardless of the strategy you use. The target environment needs to be secure. This means hardening the underlying operating systems the containers would be running on, but it may also mean establishing proper VPC and firewall rules or creating special accounts to limit access.

One possible way to limit access to the running systems is to use an orchestration system, of which the most popular today is Kubernetes. Orchestration tools often provide a secure API endpoint and Role-Based Access Control (RBAC) to minimize the risk of unauthorized access or unnecessarily high privileges.

If you deploy to cloud environments you should consider immutable deployments. This means preparing an instance image as one of the build steps. The deployment then consists of creating new instances based on this instance image. An update to the application requires creating new images, spinning new instances, then destroying the old ones.

This approach has an additional benefit: you don’t need to worry about user privileges so much. Since the hosts are immutable, there is no need for SSH access. Of course with no shell, access observability plays an even bigger role than usual. A proper monitoring and logging solution is necessary to help with debugging.

Securing Runtime

Threats may also arise during the regular operation of containers. While the previous steps are rather discreet, operating in production is continuous and requires serious planning.

Contrary to images, that we can handle in isolation, containers usually act as networked microservices. This means that the attack surface is much bigger. Creating separate virtual networks for your containers can add some depth to your defenses if one of the containers becomes compromised. Remember the principle of least privilege and only allow connectivity between those containers that actually need it. An NGINX server needs to contact the backend application, but it doesn’t need to access the database, the RabbitMQ, or Redis.

It is common knowledge that you should only expose ports that serve the application and nothing else (the SSH being the only exception). Keep this advice in mind both regarding your containers and the underlying machines they operate on. And when you open those ports to allow communication, let your services speak over TLS. The benefits of this approach are that your traffic would be encrypted (thus harder to sniff) and that you authorize your endpoints.

The principle of least privilege also applies to bind-mounted volumes. These are the volumes from the host operating system that you reference with a path, like `-v./nginx.conf:/etc/nginx.conf`. You get the best security by not using them at all. And if you must, make them read-only. Stateful containers shouldn’t require bind-mounts and would be better off with named mounts instead (`-v database:/var/db`).

During development, we talk about how to keep images secure. In runtime, the problem becomes: how to make sure we only run secure images? One answer to this question is a Docker plugin called Docker Image policy plugin. Its role is to prevent pulling any other images than those whitelisted.

And as previously mentioned, Docker’s Notary helps with ensuring the integrity of an image.

Always Be on the Lookout

The hardest thing about security is that it is not a one-time process. You cannot run a security certification to make sure your system is free of any threats. The closest you can come to achieving this is to learn that there were no threats discovered. That is different from proving that no threats exist!

Therefore to approach security with diligence is to constantly monitor your system. Log every access to your applications, services, and systems. Perform regular audits against this data. If you notice an anomaly in a user's behavior it may mean an account has been compromised. Act on it!

Monitoring similar anomalies in the network traffic can help you uncover potential attacks. There are different vectors a malicious party can use to gain unauthorized access. Logs and monitoring won't solve everything but they can be a great source of information even before an incident happens.

Remember that each day researchers find new security holes. Our systems are complex and have many dependencies. Chances are high one of those dependencies will become vulnerable during the lifetime of your application. Staying prepared for that is the best way to maintain the security of your systems.