Secrets to a Stronger Strategy For Container Security
Table of Contents

3 The fellowship of containers, an emerging developers’ community

4 Set permissions for users and system resources

6 Create an action plan based on log monitoring and ids data

8 Consider container security in the larger context

9 Meet your secret agent, alert logic

10 Deploy today through AWS marketplace
The Fellowship Of Containers, an Emerging Developers’ Community

In recent years, we have seen container adoption continue to grow as more and more organizations look to transition away from virtual machines to microservices-based architectures. Containers provide increased efficiency, portability, and scalability. Today, orchestration platforms like Docker and Kubernetes are some of the most widely adopted technologies because of the central role they play in developing with containers.

Compared to virtual machines, containers have a smaller footprint which means a smaller attack surface. They also provide an additional layer of security through their ability to isolate applications. However, this doesn’t mean that your containerized environment is not susceptible to malicious attacks between containers or within the shared resources of the underlying host.

A strong container security strategy starts with a 360-degree awareness of the container and how it interacts with its environment and ends up as automated governance policies woven into the Continuous Integration/Continuous Delivery (CI/CD) pipeline. Planning a container security strategy for the first time can feel a little like trial-and-error. And while DevOps is all about iterating to get better, security isn’t something you should take a chance on.

Our container experts have developed, tested, and refined eight best practices for security in your environment. In the spirit of the fellowship of containers, we are passing along these not-so-secret practices to you:

› Set permissions for users and system resources
› Create an action plan based on log monitoring and IDS data
› Plan for maximum portability
› Take care of your host
› Join a community forum
› Perform regular backups
› Only use trusted software
› Think big picture security

Read on for more explanation of these practices and some tips for how your Amazon Web Services (AWS) environment can help.

And welcome to the fellowship of containers.
Set Permissions For Users And System Resources

Two of the biggest differences between containers and other virtualized assets that affect your security strategy are 1) many containers share a single operating system (OS) and 2) containers ship with fully stocked libraries already built in. These are important because they create new entry points for security threats. In the case of the shared OS, or host, a single OS attack will likely affect many connected containers. Likewise, a library or binary can be used as a point of entry between containers through privilege escalation.

Thankfully, you can mitigate many of these risks by thinking about your user permissions policies and how you allocate shared resources from the get-go. Here are detailed steps to help you when you’re setting up your containers for use in your DevOps pipelines. If you’re already in production, it’s not too late to use these steps for future iterations of your microservices.

Who is Accessing Your Containers?

Set least-privilege access rules: As with any software system you run, it’s a good idea to use the lowest privileges possible for your containers, as well as the binaries and libraries within them. This helps prevent privilege escalation and wrongful data access and all kinds of bad habits.

Tip: AWS Identity and Access Management (IAM) offers a shortcut to implementing least privilege restrictions based on AWS best practices for leveraging roles established within one service to another. IAM policies can be automatically applied based on a task definition that you set when launching the container and attached to either users or roles.

Turn off root privileges: We recommend setting up a non-root user and making it the default in your container configuration. For the most part, Docker and Kubernetes subprocesses do not run with root privileges out of the box, however, it never hurts to double check. And when you absolutely must use the root account for container-based actions, be mindful of how you use it.

Tip: Amazon Elastic Container Service (Amazon ECS) provides a configuration flag that lets you choose the user you want for any task when you launch. Flag your non-root user.
Are your containers talking behind your back?

**Segregate containers:** You can lower your overall risk exposure by establishing smaller groups of containers that don’t talk to one another. Depending on your environment, you might segregate containers by host, by the role they play (i.e. web server, database, customers), or based on their risk of exposure—so those most susceptible are grouped together.

- Tip: You can set Amazon ECS to perpetuate the segregation that you establish in your initial configuration.

**Disable inter-container communication:** Another way to cut down on inter-container communication (icc) is to disable it.

- Tip: Amazon ECS lets you use a link flag to connect containers and control communications between them. You can set it up by marking docker flags “–icc=false” and “–iptables=true”.

**Restrict network chatter:** Of course, security threats can be transmitted between containers through your network, too. Be sure to consider your network controls.

- Tip: Amazon ECS offers network control through the container control for Elastic Network Interface (ENI). You can use the ENI to customize port configurations for your use case. Furthermore, AWS App Mesh allows you to standardize network communication by giving you visibility into how things are communicating on an application level.

How does your host affect your security?

**Limit host resources by container:** A denial of service (DOS) attack on a container could deplete its host’s resources and consequently shut down the other containers supported by it. By using container orchestration frameworks like Docker Swarm and Kubernetes, you can limit CPU, RAM, and ulimits for each container, which can help reduce DOS attacks and general resource hogging.

- Tip: Amazon ECS allows you to configure CPU, RAM, and ulimits to help you automate governance.

**Restrict kernel capabilities:** Set limits on what the kernel can access by task.

- Tip: With Amazon ECS, kernel capabilities are limited by the service and can be set per task.

**Validate your host before launching:** One of the biggest challenges in SecOps is incorporating repeatable container governance policies into your pipeline. You can use an automated service like Docker Bench to validate your container host against security best practices.

- Tip: In an AWS environment, you can use AWS Lambda to build Docker Bench into your CI/CD pipeline so it automatically calls the service whenever you launch a new host.

**Control file system access:** There’s no reason for the data in your file systems to be accessible to all your containers and users, but it is unless you specify otherwise. By configuring the file systems directory default to read only, you can make it so that only the host can access the data. SELinux provides a default for docker that enforces the read/execute to /usr.

- Tip: Use the Amazon ECS configuration flag to turn on –read-only.
Which resources do you need in your container?

**Remove static libraries and binaries:** Containers ship prepopulated with libraries and binaries you’ll never need to use. These can be used as a point of entry if you’re not careful. By turning on the SECCOMP modular that’s built into the Linux kernel, you can limit system calls and enable least privilege access:

- Tip: Remove setuid/setgid binaries from images
- Tip: Debian ‘defanged’ image Dockerfile configuration (removes access to those binaries)

**Store secrets separately:** As a rule of life and app building, secrets are necessary—and messy. Instead of embedding your secrets in your app configuration directly, which could compromise the security of other apps that use similar secrets, you can store secrets separately.

- Tip: AWS Systems Manager Parameter Store will store and encrypt your secrets separately from your app and its container. Systems Manager Parameter Store is fully integrated with AWS which means you can use IAM roles to call the System Manager Parameter Store to retrieve the secret for the system task (i.e. logging into the server) without embedding the secret into the app. At the application level, AWS Secrets Manager allows you to store secrets that manage application and database passwords.

Create an action plan based on log monitoring and ids data

One of the key tenets of any security strategy is to keep a close eye on what’s going on in your environment and have a plan of action for unwanted interruptions.

The same rules apply to your container environment. That’s where IDS and log monitoring come in. An intrusion detection system (IDS) gives you a holistic view of network traffic between your containers. It provides alerts on malicious traffic to keep you informed of activity. Logs, on the other hand, record forensic data you can use to get a picture of what is going on at the system level.

By using both of these together, you get a much better understanding about what is happening within a single container and between multiple containers. Not only is this important from a security perspective, but also a network and general software perspective.
How well do you understand your traffic?

**Map out container traffic:** It’s important to start with a good understanding of the traffic you expect to see traveling North/South (from container to its host) and East/West (between containers themselves) to help you better detect anomalies. You can use your network architecture or a 3rd party tool to help you map out the network pattern for expected behavior. Based on your map, it’s a good idea to disable inter-container communication (icc) that is not specifically needed (see “Are your containers talking behind your back?” for guidance).

**Monitor your traffic:** Once you know what expected traffic looks like, you need a mechanism to monitor actual traffic so you can spot traffic mishaps. That’s where IDS comes in. When evaluating your options for a 3rd party IDS solution, consider whether you need an integrated or side car solution. In an integrated solution, you run a container on the host that has access to ETH 0 bridge and can see all the traffic from the inside the system. With a side car solution, the IDS runs outside of the system and gets fed information on the containers. This often comes in the form of log correlation.

**Tip:** AWS CloudWatch allows you to configure your container instances to send log information to CloudWatch Logs. Here you can view all the different logs from your container instances in one convenient location.

Are you using the data from your logs effectively?

**Output logs to a centralized location:** If your IDS detects a problem, you can use your log data to understand what’s happening inside the environment and to perform forensics. Of course, logs can only help if you can get the data. You’ll want to configure containers so they output logs to a centralized location, ideally a separate container, where you can use a log management system to help you make sense of them. For continuous logging on docker you may have to configure the default logging driver to write logs to your desired location (/var/log/, /var/log/docker/).

**Tip:** AWS S3 is an object storage service that can be used to store the logs. AWS CloudWatch Logs is a log service to which most AWS services can output logs. CloudWatch Logs provides a single view over all the different logs from your container instances in one convenient location, allowing you to gain deeper contextual evidence on the findings, manipulate the data, and take action on it.
Consider Container Security In The Larger Context

A lot of the differentiated work that’s involved with creating a strong security strategy for your container environment, versus other virtualized assets, comes down to setting permissions, limiting resources, and monitoring systems. However, there are other considerations unique to containers and general to development environments that you need to account for as well. While guidance for those matters tends to be less prescriptive, that doesn’t make them less important.

Start Your Container Strategy With Security In Mind

Join a community forum: The fellowship of containers runs wide and deep. Joining a community forum is a great way to tap into all the expertise out there. Some popular forums include Docker, AWS, Kubernetes, as well as independent forums built around these products.

Plan for maximum portability: Your container’s portability is one of its main attributes, so make sure your security strategy (and your application itself) works across multiple platforms, in hybrid environments, as well as on-premises.

Take care of your host: A not-so-secret secret is “a healthy host is a happy host,” meaning you will save yourself a lot of heartache and work if you remember to keep your main host up to date. A great way to do that is automate regular patching using AWS Lambda and governance policies using Docker Bench as part of your CI/CD pipeline. Alternatively, you can use a managed container service, like AWS Fargate, to take care of the host for you.

Apply Security Best Practices You Already Use To Containers

Perform regular backups: This one goes hand and hand with the previous best practices. Always do regular backups and backups at important time intervals, such as before updates or any major development changes. Regular automatic updates also help in the case of disaster recovery.

Only use trusted software: It may be tempting to (after reading a good article on a random blog or receiving a link) pull an image from an unknown repository. Don’t. If you can’t find the images from a trusted source there’s probably a good reason.

Think big picture security: A strong security strategy requires a thorough understanding of how everything works together. Whether you’re using containers for development or running production servers for ecommerce, it’s a good idea to start by outlining your goals and security posture. You can do that by creating an architecture diagram that lists how everything operates together and using that hierarchy to apply the model of least privilege. Furthermore, you can implement pre-production scanning of your containers as part of your CI/CD pipeline. Creating a Wordflow diagram makes it easier to understand the process and automate the scanning where appropriate. (Note: Alert Logic products include scanning capabilities you can build into this process.)
Meet Your Secret Agent, Alert Logic

At Alert Logic, our contingent of the fellowship of containers has created services specifically designed to help you build a strong container security strategy without requiring you to get your PhD in it.

All of our offerings employ our AWS cloud-native security platform, threat intelligence with prioritized remediation recommendations, and use our 24/7 expert defender team. Our most popular offering, Alert Logic Professional, includes a fully managed service in which our experts monitor your environment 24/7.

Unlike many of the other solutions available today, ours has been developed to work with AWS native architecture as well as other cloud and on-premises systems. We are certified as an AWS Container Competency partner.

Alert Logic Essentials uses SIEMless Threat Management™ to help you gain visibility into your environments, identify appropriate remediation steps, and block attacks through a combination of machine-learning and real-time behavior analysis.

- **Asset Discovery**
- **Vulnerability Scanning**
- **Cloud Configuration Checks**
- **Extended Endpoint Protection**
- **Threat Risk Index**
- **Compliance Scanning And Reporting**
- **Support For Multiple Environments**
- **24/7 Email And Phone Support**

Alert Logic Professional includes all the benefits of Essentials plus these capabilities, and more:

- **24/7 Incident Monitoring & Management**
- **Security Analytics & Threat Intelligence**
- **Log Collection and Monitoring**
- **Intrusion Detection System**
- **Security Event Insights and Analysis**
- **Office 365 Log Collection & Search**
- **Cloud Vendor Security Integrations**
- **User Behavior Anomaly Detection**
- **Anti-Virus Integration**
Lock In Your Container Strategy With Alert Logic

Subscribe to Alert Logic Managed Threat Detection on AWS Marketplace to get started fast.

SIGN UP NOW