Component: Container Runtime

Table of Contents

Overview

This page provides a detailed design for the Container Runtime Component of the COPS Platform. It is based on the Decision from this Decision Log: Which container runtime are we using for COPS?

In the future, this may be updated to use Firecracker MicroVMs instead of Containers. For now, Docker is the best approach to get a prototype up and running.

Responsibilities

- Provide compute resources for RESTful services behind the API Gateway
- Authenticate users with an IdAM service
- Ensure that only authorized users can access each container
- Start containers with specific access controls when requested by the API Gateway
- Preferably use SELinux labeling to enforce access control

Overall Strategy

Docker provides the central functionality for this component.

Component Interfaces

Incoming connections on TCP Port 2376 (API Gateway requests for container startup).

Outgoing connections to TCP Port 514 (Logs sent to rsyslog)

Incoming connections on TCP Port 443 (and others) (Services running inside the Container Runtime)

Configuration

SELinux

SELinux must be enabled and in ‘enforcing’ mode on the host.

Docker

Docker must be installed on the same CentOS 7 host that will be running the Data Storage Component (due to the dependency on SELinux). The other components could potentially be running elsewhere, but the Container Runtime and Data Storage Components need to be on the same host.

General Installation and Configuration

The Securing Docker with SELinux page provides general guidance for configuring Docker to use SELinux for Container isolation. Follow the steps to enable the Docker SELinux Policy.

While following that documentation, Docker must be configured as follows:

- Allow connections from the API Gateway to the Docker Daemon. See the implementation details of the API Gateway for a final answer, but this likely means that TCP Port 2376 should be configured for encrypted communication.
- Docker containers must be run with the security levels and categories of the user requesting the service. The API Gateway sends the request to the Docker Daemon with the values.
    - In particular, the request must set the "--security-opt label=…" option when starting the container.

Logging and Auditing

All logs must be sent to the external Log Aggregator (currently expected to be rsyslog).

References

- Docker Logging configuration: https://docs.docker.com/config/containers/logging/configure/
- Configuring socket communication for the docker daemon: https://docs.docker.com/engine/reference/commandline/dockerd/
- Setting the SELinux label when starting a docker container: https://docs.docker.com/engine/reference/run/ (search "Security configuration")

Expectations of services running inside containers

General

- At any time, multiple instances of the service will be running inside different containers. They must not expect to share information between themselves. Each instance will handle requests from one or more users with identical SELinux security levels and categories.
- Startup should be completed as quickly as possible. The docker images should include all executables, libraries, and configuration data so that application startup is the the only action required when the container starts.
- If appropriate, the Service may authenticate users against the IdAM service. The API Gateway will have already validated the user authentication and authorization, and will have effectively sandboxed the service into a container with restrictions based on the user's access levels.

Handling Labeled Data

- Each container will be labeled with the security levels and categories associated with the user that is requesting the service. As such, the service must gracefully handle 'Access Denied' errors when attempting to retrieve data from the Data Storage component.
- Each service will be given login credentials to connect to the SE-PostgreSQL database within the Data Storage component. Those credentials allow the initial connection, but the added layer of SELinux labels mean that each instance of the service (running inside different containers) will have access to different information. The service must be written to expect this.

Nonfunctional requirements for the software

- The installation procedure must work in an offline/disconnected environment.
- The final configuration must run on CentOS 7 with SELinux enabled (and enforcing)