What is "privileged mode" and do I need it?

Introduction

This blog post is written for software developers who understand basic concepts about containerized applications (hereinafter referred to as containers), and are new to using Open Horizon to deliver edge computing services.

On a host machine, some tasks can only be performed by an account with root access. This means that the account you are currently logged in as is either the root account itself (generally not a good idea), or your account has acquired root-level privileges through `sudo`. Likewise, containers generally do not need privileged mode on the host: to be run as the root user or to have root-level access on the host computer.

In Open Horizon and all commercial distributions based on it, you have the ability to specify that a service should be deployed with privileged process execution enabled. By default, it is disabled. You must explicitly enable it in the respective Service Definition file for each container that needs to run in this mode. And further, any node on which you want to deploy that service must also explicitly allow privileged mode containers.

The reason for requiring the node policy file to explicitly enable privileged mode is because the node owner gets a say/vote in what runs on the node. This is the whole purpose of the node policy, to give the node owner agency in the decision about what runs there. If the service definition ***or one of its dependencies*** requires privileged mode, the node policy must also allow privileged mode, **or else services will not be deployed to the node**.

How does privileged process execution impact security?

A major security principle the Open Horizon project follows is: "***All parties are untrusted by default.***" As a result, Node Policies and Service Definitions do not allow privileged process execution by default. You must explicitly enable it in both the node and the service if you want to deploy and run a service that requires it.

However, a privileged container is a powerful and potentially dangerous tool and should not be used without considering alternatives. If you run a container with privileged access, it can access all resources on the host system as the root user. If a privileged container can be hacked by a third party, that third party could then gain access to all resources on the host computer.

Therefore, try not to use privileged containers. If you must, use the following guidelines to ensure that privileged containers:

- are thoroughly and continuously vetted for vulnerabilities
- have a narrow scope for their duties ... meaning they should only perform a specific task
- only mount necessary host directories and devices (Specified in the Service Definition file. See the example at the bottom of this blog post.)

Why do we use it (what is it good for)?

With all of the potential drawbacks, what situations require a container to run as privileged?

1. Does your code require direct access to host hardware? For example, you may need to use a microphone in order to record and analyze sound waves. You might need to use the host GPU for model (re)training. You could potentially need to access a video stream directly from an attached camera. In these situations, you should first try to bind mount the device to see if that approach is sufficient. Another approach is to use `cap-add` to add only the kernel capabilities that you specifically need. By way of contrast, privileged mode adds *all* of the kernel's `CAP_*` capabilities.
2. Does your service need to spawn other containers? This is a common task in CI/CD pipelines. You may be able to spawn containers by bind mounting the docker daemon socket (normally /var/run/docker.sock) in systems that use docker. In systems that use podman, which has no daemon process, you may need elevated privileges. In any case, if your container can spawn other containers without restrictions then it can effectively run any code as root on the host.

When should you not use it?

If you need to access a file on the host with elevated permissions, try mounting the file into the container and ensuring that the container is running as a user that belongs to the same group as the file owner, with group read privileges enabled. Do not run the container as privileged just to read from or write to a file.

Can you show me an example?

The `audio2text` example service requires access to the microphone to record audio, and thus needs privileged access to the host hardware. View the Service Definition's deployment section to see how it is enabled.