Podman with capabilities on Fedora

By shiwanibiradar on November 16, 2020 12

Containerization is a booming technology. As many as seventy-five percent of global organizations could be running some type of containerization technology in the near future. Since widely used technologies are more likely to be targeted by hackers, securing containers is especially important. This article will demonstrate how POSIX capabilitiesare used to secure Podman containers. Podman is the default container management tool in RHEL8.

Determine the Podman container's privilege mode

Containers run in either privileged or unprivileged mode. In privileged mode, the container uid 0 is mapped to the host's uid 0. For some use cases, unprivileged containers lack sufficient access to the resources of the host machine. Technologies and techniques including Mandatory Access Control (apparmor, SELinux), seccomp filters, dropping of capabilities, and namespaces help to secure containers regardless of their mode of operation.

To determine the privilege mode from outside the container:

```
$ podman inspect --format="{{.HostConfig.Privileged}}" <container id>
```

If the above command returns *true* then the container is running in privileged mode. If it returns *false* then the container is running in unprivileged mode.

To determine the privilege mode from inside the container:

```
$ ip link add dummy0 type dummy
```

If this command allows you to create an interface then you are running a privileged container. Otherwise you are running an unprivileged container.

Capabilities

Namespaces isolate a container's processes from arbitrary access to the resources of its host and from access to the resources of other containers running on the same host. Processes within *privileged* containers, however, might still be able to do things like alter the IP routing table, trace arbitrary processes, and load kernel modules. Capabilities allow one to apply finer-grained restrictions on what resources the processes within a container can access or alter; even when the container is running in privileged mode. Capabilities also allow one to assign privileges to an unprivileged container that it would not otherwise have.

For example, to add the *NET_ADMIN* capability to an unprivileged container so that a network interface can be created inside of the container, you would run *podman* with parameters similar to the following:

```
[root@vm1 ~]# podman run -it --cap-add=NET_ADMIN centos
[root@b27fea33ccf1 /]# ip link add dummy0 type dummy
[root@b27fea33ccf1 /]# ip link
```

The above commands demonstrate a *dummy0* interface being created in an unprivileged container. Without the *NET_ADMIN* capability, an unprivileged container would not be able to create an interface. The above commands demonstrate how to grant a capability to an unprivileged container.

Currently, there are about 39 capabilities that can be granted or denied. Privileged containers are granted many capabilities by default. It is advisable to drop unneeded capabilities from

privileged containers to make them more secure.

To drop all capabilities from a container:

```
$ podman run -it -d --name mycontainer --cap-drop=all centos
```

To list a container's capabilities:

```
$ podman exec -it 48f11d9fa512 capsh --print
```

The above command should show that no capabilities are granted to the container.

Refer to the *capabilities* man page for a complete list of capabilities:

```
$ man capabilities
```

Use the *capsh* command to list the capabilities you currently possess:

```
$ capsh --print
```

As another example, the below command demonstrates dropping the *NET_RAW* capability from a container. Without the *NET_RAW* capability, servers on the internet cannot be pinged from within the container.

```
$ podman run -it --name mycontainer1 --cap-drop=net_raw centos
>>> ping google.com (will output error, operation not permitted)
```

As a final example, if your container were to only need the *SETUID* and *SETGID* capabilities, you could achieve such a permission set by dropping all capabilities and then re-adding only those two.

```
$ podman run -d --cap-drop=all --cap-add=setuid --cap-add=setgid fedora sleep 5
> /dev/null; pscap | grep sleep
```

The *pscap* command shown above should show the capabilities that have been granted to the container.

I hope you enjoyed this brief exploration of how capabilities are used to secure Podman containers.