What is the Docker socket?

The Docker socket (`/var/run/docker.sock`) is the API endpoint for the Docker daemon, which runs as root on the host. Anything that can talk to it can create containers, mount the host filesystem, and run code as root on the host. Mounting the socket into a container (a common convenience for CI and monitoring tools) therefore hands that container full control of the host, making escape trivial. Treat socket access as root-equivalent.

HOW IT WORKS

01 The abuse and payload

If a container has the socket mounted (`-v /var/run/docker.sock:/var/run/docker.sock`), an attacker inside uses it to own the host:

- Install the docker client or use the API directly, then launch a container that mounts host root:

```
docker -H unix:///var/run/docker.sock run -v /:/host -it alpine chroot /host sh
```

- That gives a root shell on the host filesystem, escaping the original container entirely.

The same is true for an exposed TCP Docker API (port 2375 without TLS). Documented techniques shown for defenders.

> SOCKET = ROOT ON HOST
>
> *Access to* `docker.sock` *is equivalent to root on the host, full stop. Mounting it into a workload is one of the most common ways a single compromised container becomes a host takeover.*

HOW TO DEFEND

- Do not mount the Docker socket into application containers. Find another way to do what the tool needs.
- Never expose the Docker API over TCP without mutual TLS; avoid 2375 entirely.
- Use rootless Docker or a socket proxy that allows only the minimal API calls a tool requires.
- In Kubernetes, prefer the standard APIs over mounting host sockets; block hostPath mounts of sockets.
- Scan for socket mounts in compose files and manifests.

SOURCES

[1] Docker docs: Daemon remote access

[2] NIST SP 800-190 Application Container Security Guide

[3] MITRE ATT&CK: Containers Matrix

Find the container escape paths before an attacker does.

`securelayer7.net/learn/containers/what-is-the-docker-socket`

Open online