A Docker vulnerability is any flaw or weakness in a Docker host, container, or image that could be exploited.

Some general Docker vulnerabilities include:

1. Having host/Docker not up to date. Some vulnerabilities appear in older versions of Docker, such as:
    a. **Dirty pipe**: A pipe is a way to communicate data from one app or process to another on Linux. However, a flaw in this kernel pipeline implementation allows an attacker to manipulate Linux pipes, overwrite the content of files from container images that they don't have permission to change, and then escalate their privileges locally to get full system access.
    b. **Improper initialization:** In certain Docker versions, code injection by malicious actors can occur when the nsswitch (name service switch) dynamically loads a library inside a chroot (change root) that contains the contents of the container. An application can then execute code from an attacker.
    c. **IPv6 input validation vulnerability**: In Docker Engine, attackers can execute code in a container to spoof rogue IPv6 router advertisements, thus performing a man-in-the-middle attack (MitM) against another container or the host's network. To mitigate, the solution is to prevent untrusted, non-privileged containers from running with CAP_NET_RAW.
2. User privilege escalation. Often, attackers can escalate their privileges to the entire host and gain root access. Configuring the container to use an unprivileged user, not root, is the best way to prevent these attacks, such as:
    a. **runC vulnerability**: runC is a standard runtime for spawning and running containers. However, if an attacker has root access in a container where runC is tasked with running a user-defined binary in the container, the attacker can overwrite the host and escalate their privileges. They can now execute runC and achieve code execution on the host, gaining root access on the host.
    b. **Container escape** vulnerabilities could occur where adversaries can escape containers and perform arbitrary command execution on the host machine, elevating their privileges.
3. Privileged containers: A privileged container has all the capabilities a host can perform, allowing attackers to gain access. Only specific capabilities needed by a container should be granted, and containers should not be run using the –privileged flag.
4. Exposing the Docker daemon socket, which is equivalent to giving unrestricted root access to your host.
5. Not limiting resources. It is very important to limit memory, CPU, maximum number of restarts, file descriptors, and processes using ulimit, otherwise vulnerabilities can occur that crash the system, such as:
    a. **Improper Check for Unusual or Exceptional Conditions:** In older Docker versions, pulling an intentionally malformed Docker image manifest will trigger resource exhaustion and crash the Docker daemon running on the host system.

To avoid many of these vulnerabilities, some best practices include regularly updating Docker and the host so vulnerabilities are patched. Furthermore, downloading images from trusted sources only and running Docker containers as non-root helps to prevent malicious content from accessing the host container and ensuring the application environment is safe. Container scanning for the container's software, interactions with host system and other containers, and configurations also detects and prevents vulnerabilities.