


DOCKER SECURITY CHEAT SHEET

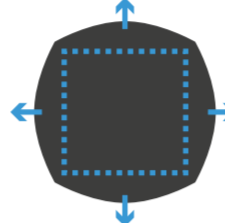
DISCLAIMER: The following tips should help you to secure a container based system. They are not a complete solution and will not in themselves guarantee security. They should only form a small part of your security policy which should mandate a holistic approach with multiple layers of defence.

For more in depth information about Docker security please refer to: **"Using Docker. Developing and Deploying Software with Containers"**
By Adrian Mouat, Publisher: O'Reilly Media


TYPES OF SECURITY THREATS AND HOW TO AVOID THEM




KERNEL EXPLOITS
If a container can cause a kernel panic or similar, it will bring down the whole host.




DENIAL OF SERVICE (DOS) ATTACKS
All containers share kernel resources. If one container monopolizes access to a resource, it will starve out the other containers.



CONTAINER BREAKOUTS
If an attacker can breakout of a container, they can gain access to the host and other containers.



POISONED IMAGES
Images may be injected with trojan or virus infected software. Or they may simply be running outdated, known-vulnerable versions of software.



COMPROMISED SECRETS
API keys and database passwords must be kept secure to prevent attackers gaining access.

Tip	Kernel Exploits	DOS Attacks	Container Breakouts	Poisoned Images	Compromised Secrets
SEGREGATE CONTAINER GROUPS WITH VMs		○			
DEFANG SETUID/SETGID BINARIES	○		○		
BE AWARE OF CPU SHARES		○			
VERIFY IMAGES				○	
SET CONTAINER FILE SYSTEM TO READ-ONLY	○	○	○		○
SET A USER	○		○		○
DO NOT USE ENVIRONMENT VARIABLES TO SHARE SECRETS					○
DO NOT RUN CONTAINERS WITH THE --privileged FLAG	○		○		○
TURN OFF INTER-CONTAINER COMMUNICATION	○	○	○		
SET VOLUMES TO READ-ONLY	○		○		
SET MEMORY LIMITS		○			
DO NOT INSTALL UNNECESSARY PACKAGES IN THE CONTAINER	○		○		

SEGREGATE CONTAINER GROUPS WITH VMs

Keep containers belonging to different users, or operating on sensitive data, in separate VMs. This will limit the damage if a container breakout occurs.

DEFANG SETUID/SETGID BINARIES

setuid/setgid binaries run with the privileges of the owner and can sometimes be exploited by attackers to gain elevated privileges. To find such binaries:

```
$ docker run debian \
  find / -perm +6000 -type f -exec ls -ld {} \; 2> /dev/null
-rwsr-xr-x 1 root root 10248 Apr 15 00:02 /usr/lib/pt_chown
-rwxr-sr-x 1 root shadow 62272 Nov 20 2014 /usr/bin/chage
-rwsr-xr-x 1 root root 75376 Nov 20 2014 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 53616 Nov 20 2014 /usr/bin/chfn
...
```

To disable setuid rights, add the following to the Dockerfile

```
FROM debian:wheezy
RUN find / -perm +6000 -type f -exec chmod a-s {} \; \
  || true
```

BE AWARE OF CPU SHARES

By default, all containers get an equal proportion of CPU cycles. This proportion can be modified by changing the container's CPU share weighting relative to the weighting of all other running containers. Containers are assigned a weighting of 1024 by default.

```
$ docker run -d -c 512 myimage
```

VERIFY IMAGES

Be careful about using 3rd party images. Only use images from automated builds with linked source code. Consider building images yourself rather than pulling from the Hub, or pulling by digest to take advantage of checksum validation e.g:

```
$ docker pull debian@sha256:a25306f3850e1bd44541976aa7b5fd0a29be
```

SET CONTAINER FILE SYSTEM TO READ-ONLY

Unless you need to modify files in your container, make the filesystem read-only.

```
$ docker run --read-only debian touch x
touch: cannot touch 'x': Read-only file system
```

SET A USER

Users in docker are not namespaced. Don't run your application as root in containers.

```
RUN groupadd -r user && useradd -r -g user user
USER user
```

DO NOT USE ENVIRONMENT VARIABLES TO SHARE SECRETS

Environment variables are easily leaked when debugging and exposed in too many places including child processes, linked containers and docker inspect. Instead consider using volumes to pass secrets in a file or using a key-value store such as etcd, Vault or Keywhiz.

DO NOT RUN CONTAINERS WITH THE --privileged FLAG

The --privileged flag gives all capabilities to the container, and it also lifts all the limitations enforced by the device cgroup controller.

TURN OFF INTER-CONTAINER COMMUNICATION

By default, unrestricted network traffic is enabled between all containers on the same host. Run containers with --icc=false --iptables in order enable only communication between containers linked together explicitly.

```
$ docker -d --icc=false --iptables
```

SET VOLUMES TO READ-ONLY

Unless you need to modify files in attached volumes, make them read-only.

```
$ docker run -v $(pwd)/secrets:/secrets:ro debian touch /secrets/x
touch: cannot touch '/secrets/x': Read-only file system
```

SET MEMORY LIMITS

Limit the max amount of memory a container can use, don't let a memory hog starve your other containers.

```
$ docker run -m 512m myimage
```

DO NOT INSTALL UNNECESSARY PACKAGES IN THE CONTAINER

Do not install anything that is not relevant to the purpose of container, including tools like ssh. Run:

```
$ docker exec $INSTANCE_ID rpm -qa
or
$ docker exec $INSTANCE_ID dpkg -l
```

to see what packages are installed in a container.