# Container Processes without Root

```
# cat Dockerfile
FROM centos:7
RUN yum -y update
RUN yum –y install httpd
RUN yum –y install net-tools
EXPOSE 8000
# docker build -t webprecursor .
# docker run --name=ctr  –it webprecursor /bin/bash
    # chown -R apache /etc/httpd/ /var/run/httpd/ /var/log/httpd/
    # vi /etc/passwd (give apache a shell)
    # vi /etc/httpd/conf/httpd.conf (change port to 8000)
# docker commit ctr web_unpriv_ctr
# docker rm -f ctr
# docker run -u apache -d -p 80:8000 web_unpriv_ctr /usr/sbin/apachectl -D FOREGROUND
```

## Running the Pods Containers as Non-root

```
apiVersion: v1
kind: Pod
metadata:
  name: container-as-non-root
spec:
  containers:
  - name: mycontainer
    image: myimage
    securityContext:
      runAsUser: 1000
      runAsGroup: 1000
```

# Docker Root Capabilities

Docker drops all root capabilities except:

| | |
|---|---|
| **CHOWN:** | Make arbitrary changes to file UIDs and GIDs (see **chown**(2)). |
| **DAC_OVERRIDE:** | Bypass file read, write, and execute permission checks |
| **FSETID:** | Don't clear Set-UID and Set-GID bits when a file is modified |
| **FOWNER:** | Bypass perm checks on operations, set ACLs, … |
| **MKNOD:** | Create special files using **mknod**(2) |
| **NET_RAW:** | Use RAW and PACKET sockets; bind to any address for transparent proxying. |
| **SETGID:** | Make arbitrary manipulations of process GIDs |
| **SETUID:** | Make arbitrary manipulations of process UIDs |
| **SETFCAP:** | Set file capabilities. |
| **SETPCAP:** | Set process capabilities. |
| **NET_BIND_SERVICE:** | Bind a socket to Internet domain privileged ports (<1024). |
| **SYS_CHROOT:** | Use **chroot**(2). |
| **KILL:** | Bypass permission checks for sending signals (see **kill**(2)). |
| **AUDIT_WRITE:** | Write records to kernel auditing log. |

## Observe a Dropped Capability

Start a root container.  Try an iptables command.

## Dropping More Capabilities

You can control what capabilities Docker retains from these, or add to these, by using `docker run --cap-add` and `--cap-drop`.

This would drop all capabilities except `net_bind_service`, which lets us bind to a privileged (<1024) port.

```
docker run --cap-drop ALL --cap-add net_bind_service image /bin/bash
```

Bonus: try running the Apache container as root, but with the minimal set of capabilities.

## Capabilities Documentation

To read more about Linux capabilities, consult:

```
man 7 capabilities
```

Here's a great article on Linux Capabilities that shows you how to use capsh to explore dropping capabilities.

https://linux-audit.com/linux-capabilities-101

## Exercise: Capability-based PrivEsc

**Advanced Privilege Escalation with Linux Capabilities**

`http://127.0.0.1:10000/morpheus-adv-privilege-escalation`

## Running the Pods Containers with Capability Dropping

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-seccomp-profile
spec:
  containers:
  - name: mycontainer
    image: myimage
    securityContext:
      runAsUser: 1000
      runAsGroup: 1000
      seccompProfile:
        type: Localhost
        localhostProfile: profile-allow.json
      capabilities:
        drop: [ "all" ]
        add: ["CAP_NET_RAW"]
```