Secure Your Apps in Production using Mesos Containerizer

HELLO!

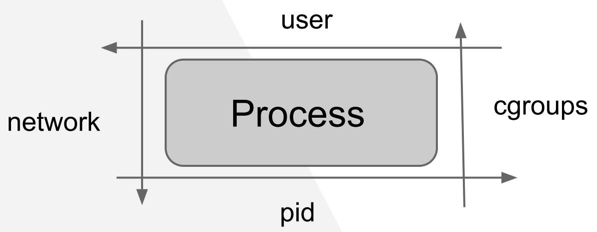
I am Benjamin Bannier
I am here because I love Containers and Mesos.
You can find me at @benjamin

Introduction

Why Containerization ?

Containers are not VMs

Containers allow you to run a linux process within certain constraints.
Isolate



Why containerization

Abstracts away underlying system
- For users
- For containerized Applications

Isolation - resources, networking and visibility
Helps to define application surface
Relevance to Enterprise

Introduction

Limits of Containerization ?

- cross talk between containers and host processes (-> seccomp)
- containers requiring privileged access to own container (-> user namespaces)
- containers requiring priviledged access to host facilities (-> capabilities)

Goals

- improved isolation
- reduce the surface area of attack
- less privileged process

User Namespaces

HELLO!

I am Srini Brahmaroutu
I am from IBM, learning Containers and Mesos.
You can find me at @srbrahma

User Namespaces

- History
- What are User Namespaces
  - Virtualize users
  - Run unprivileged containers
- Why User Namespaces
  - Protect global resources
  - Contain application’s root privileges

User Namespaces

- Mesos Tasks
  - unprivileged tasks
- Enable User Namespace on Mesos
  - Agent flags
  - Isolators
  - User mapping

User Namespaces

- Mesos Agent flags - switch_user,userns?
  - unprivileged tasks
  - Tasks running in user namespace

```
sudo GLOG_v=2 ./bin/mesos-agent.sh  --master=127.0.0.1:5050 --image_providers=APPC,DOCKER

--isolation=namespaces/user --switch_user=true &


UnprivilegedUser$> mesos-execute …  // run your task
```

User Namespaces

- Mesos Isolators for User Namespace

Create : Creates a isolator class ...

Prepare : Sets the clone flag

Isolate : Writes map file

Update/Recover/Cleanup : Not required

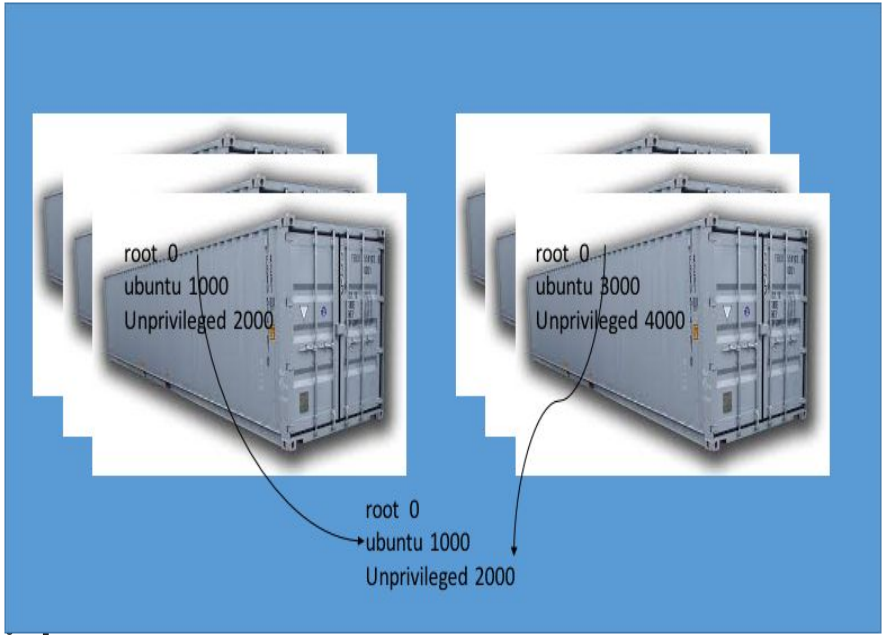
User Namespaces

- User Mapping

/proc/[pid]/uid_map

/proc/[pid]/gid_map

/etc/subuid & /etc/subgid



User Namespaces

- File sytems and User Namespaces
  - Share image layers
  - Mount filesystem

HELLO!

Again, let’s talk about Capabilities.

Capabilities

A POSIX/Linux mechanism to *divide privileges* (e.g., of root) into *fine-grained capabilities*.

Examples:

- binding to privileged ports < 1024,
- sending signals to arbitrary processes,
- bypass file permission checks,
- and many more.

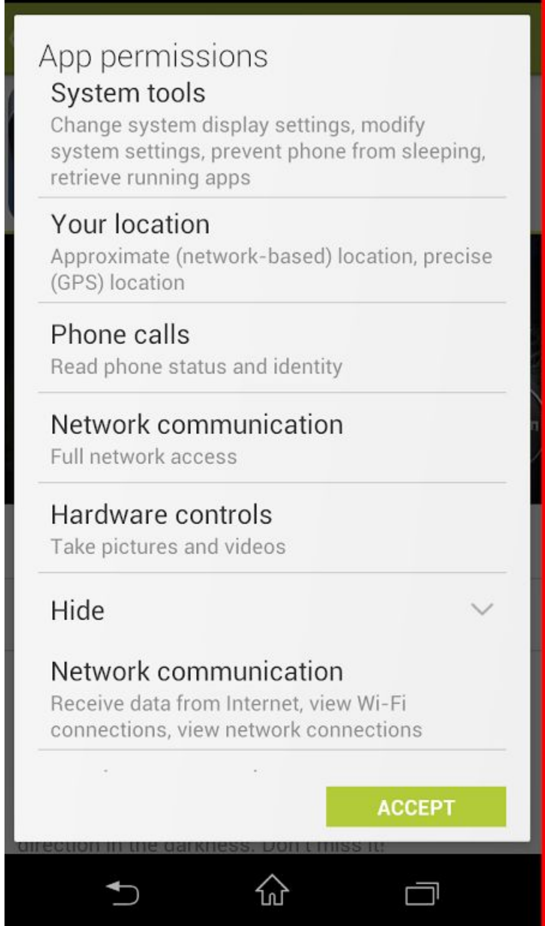
Purpose

To perform *any privileged action*, tasks needed to be run with *full superuser privileges*.

- hard to control privilege access,
- user errors can have (unintended) effects beyond their environment.

<u>Does not fit expectations for containerization well.</u>

The Competitor's Permissions



App permissions

System tools
Change system display settings, modify system settings, prevent phone from sleeping, retrieve running apps

Your location
Approximate (network-based) location, precise (GPS) location

Phone calls
Read phone status and identity

Network communication
Full network access

Hardware controls
Take pictures and videos

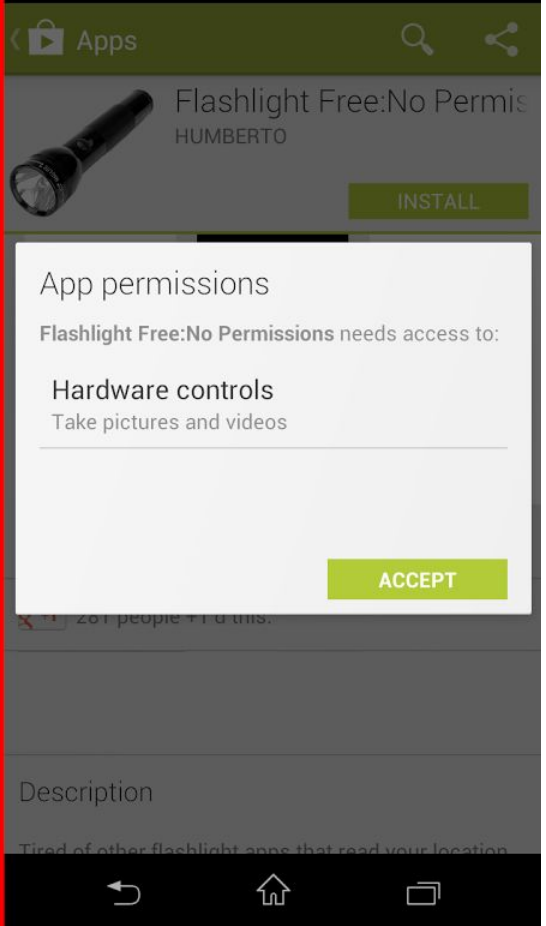
Hide

Network communication
Receive data from Internet, view Wi-Fi connections, view network connections

ACCEPT

direction in the darkness. Don't miss it!

Our App's Permissions



Apps

Flashlight Free:No Permis
HUMBERTO

INSTALL

App permissions

Flashlight Free:No Permissions needs access to:

Hardware controls
Take pictures and videos

ACCEPT

281 people +1'd this.

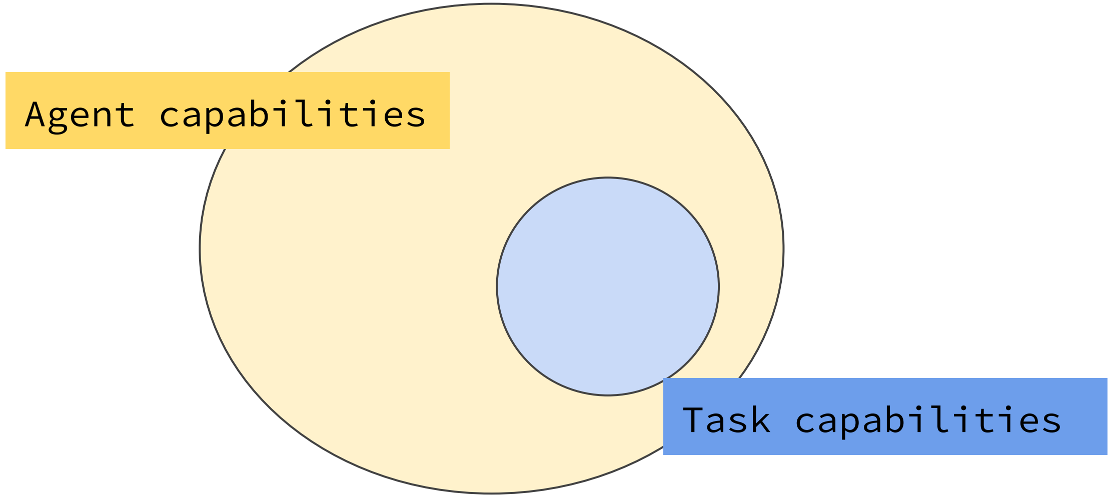
Description

Tired of other flashlight apps that read your location

Integration into Mesos

Capabilities isolator linux/capabilities.

- Operator sets up agents with set of allowed capabilities
- User request required capabilities for their tasks.



Possible future extensions

Non-root tasks can effectively only use *file-based capabilities*.

Linux > 4.3 introduces *ambient capabilities* to address this.

We could extend support for capabilities for non-root tasks, e.g., via ambient capabilities, or user namespaces.

In the context of the Mesos containerizer we introduced

- new *Mesos abstractions* for capabilities,
- interfaces for operators to grant capabilities to tasks,
- interfaces for users to request capabilities.

This adds new containerization tools for privileged tasks.

HELLO!

I am Jay Guo
I am from IBM, contributing to many open sources and Mesos.
Me: @guoger

Seccomp – What is it?

- A mechanism to restrict syscalls a process can make
- One-way transition into “secure” state.

Seccomp – Why do we need it?

- Reduce attack surface of Kernel, which is shared among containers and host.
- Execute customer’s code with more confidence.

Seccomp - How does it work?

- A Berkely Packer Filter(BPF) program loaded into kernel to control which system calles are permitted.
- Every syscall goes through the filter first
- Actions include
  - KILL,
  - TRAP,
  - ERRNO,
  - TRACE,
  - ALLOW

Seccomp – Who's using it?

- openSSH
- vsftpd
- Chrome/Chromium
- Docker
- …
- …

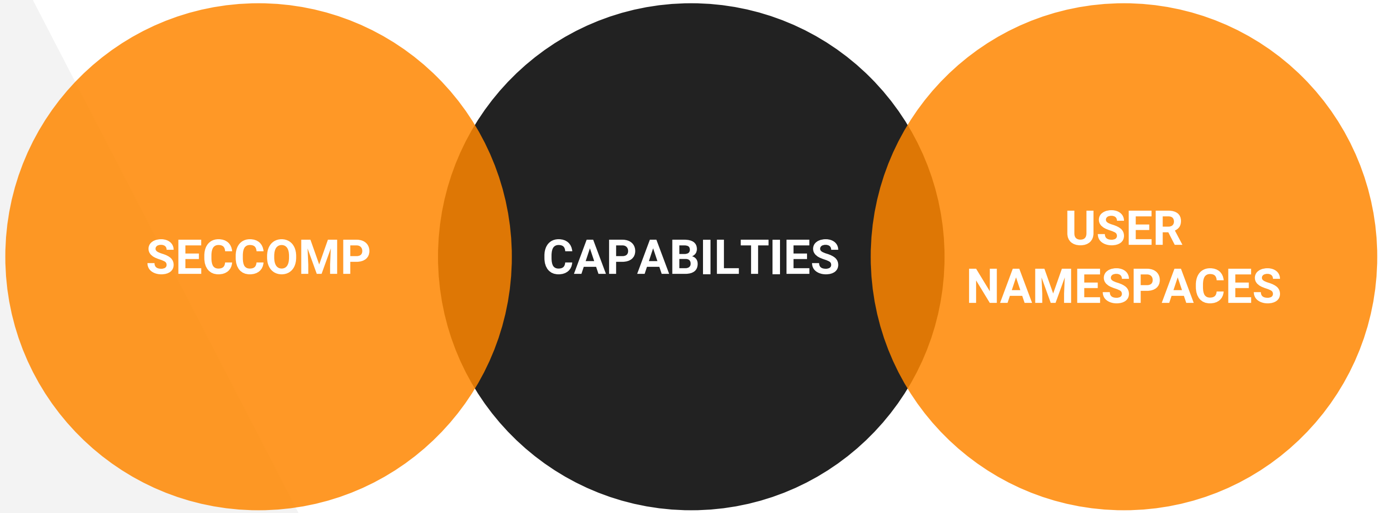
Seccomp - When it comes to Mesos ...

- Enforced by operator via mesos agent flags
  - --isolation=linux/seccomp
  - --seccomp_profile=/home/myseccomp.json
- Customized profile or default one providing mild protection.
- Stack up seccomp profiles for extra security

What can be done now ?

- User namespaces
  - Review for patches
  - Need to think about filesystems
- Capabilities
  - In the code base, use it and thrive
- Seccomp
  - Review for patches

Improved Container Security



THANKS!

Any questions?

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Startupstockphotos