Container Security From the Bottom Up

BSidesSLC 2021

Rion Carter
Jacob Carter

About

Rion Carter

Went to school for electronic engineering, wound up in technology and software. Loves to find and fix programs. Currently works as a Staff Engineer at VMware. On the weekends it’s been said he likes to bake desserts. DEF CON 29 speaker.

Jacob Carter

Took college courses in drafting, aviation, computer engineering, and then ended up in software out of interest. Currently works as an AppSec engineer at Domo. In his spare time he designs and builds replacement components for vintage Apple computers.

Agenda

0x00 - Lab Setup

0x01 - Introduction

- Marketing & Selling ‘Containers’
- Current state of the ecosystem
- Yes… but what underpins it all?

0x02 - Overview of Isolation

- Thirty-thousand Foot View
- What we’re covering today
- What we’re not covering today

0x03 - Linux Namespaces

- What are they and how to use them
- Lab work to exercise your skills

0x04 - cgroups

- Purpose/function… and labs

0x05 - seccomp

- A story of sycalls… and labs

0x06 - In closing…

0x07 - References & Tips

0x00 - Environment setup

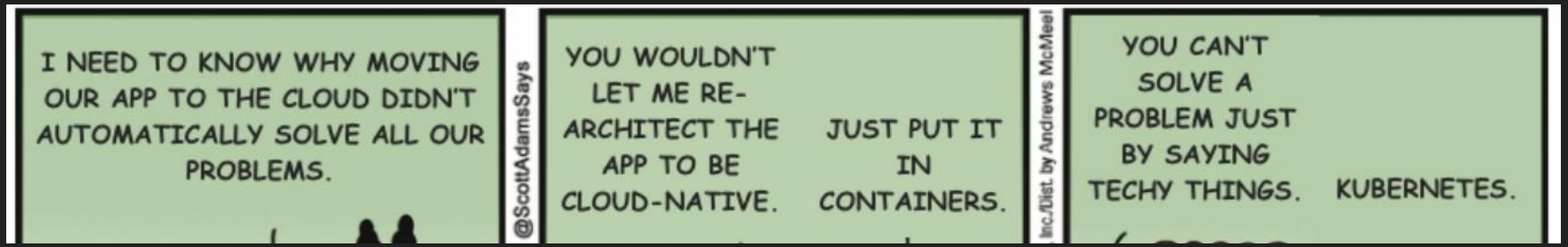
For those who did not set up the machine in advance… or those who have issues/questions, we will spend time to get you setup

https://bit.ly/2ZQ66Mk

Items:

- VMware installed
- VM installed/unpacked
- Login/can you shell?

0x01 - Introduction



0x01 - Introduction

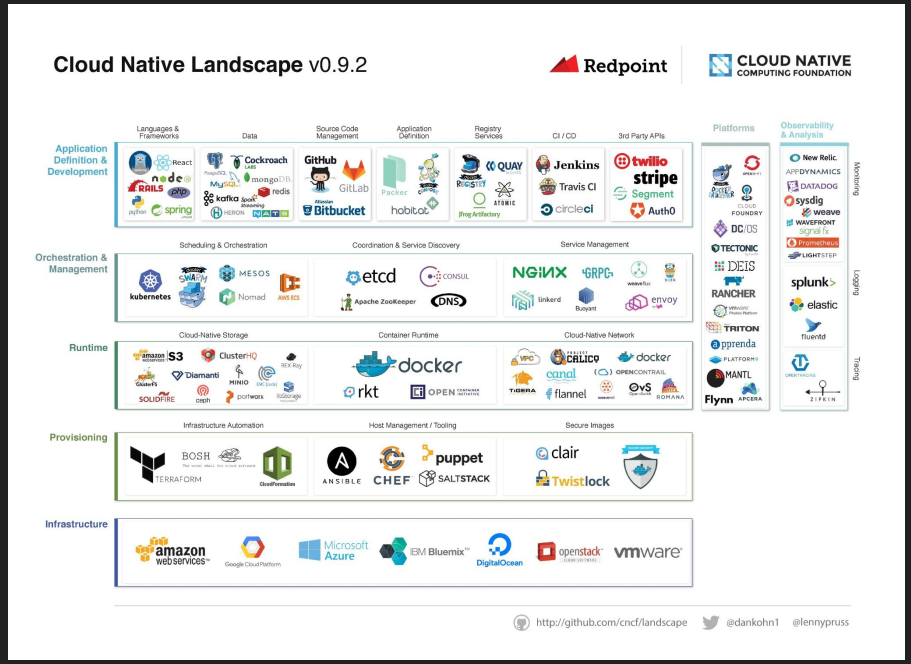
- Containerization is a ‘marketing term’ that is more of a convention of use
- Docker popularized the current ‘de-facto’ standard
- Robust, fast-moving ecosystem
- Widely used and not fully understood under the hood
  - Which is what brings us here today

Image Credits Last Slide

0x01 - Introduction

Current state of the ecosystem:

- CNCF foundation
  - Sets and promotes standards
  - ‘Owns’ many popular container OSS projects
- Containerd
  - Manages container runtime lifecycle on hosts
- Docker
  - De-facto standard for building images
- Kubernetes
  - De-facto standard for container orchestration
- Others
  - AWS ECS, podman, mesos, lxd, etc...



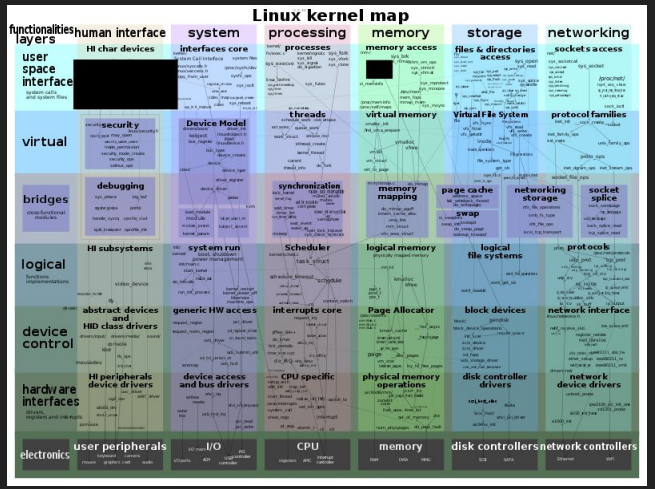
Intentionally too small to see.
There’s a lot here.

0x01 - Introduction

What underpins the ecosystem?

Linux resource isolation technologies:

- Namespaces
- CGroups
- Seccomp
  - eBPF
- Capabilities
- Linux Security Modules (LSM)

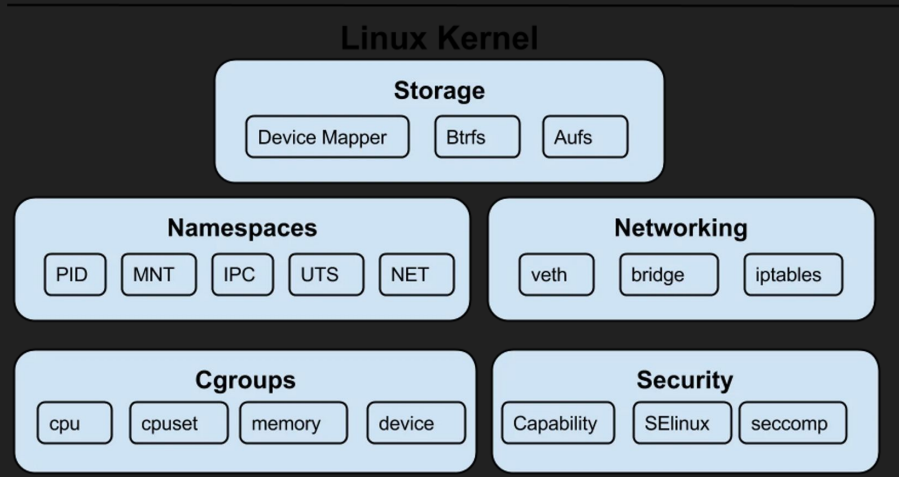


This is a seriously cool SVG

0x02 - Overview of Isolation

Linux Isolation technologies:

- Namespaces
- CGroups
- seccomp-bpf
- Capabilities
- Linux Security Modules (LSM)



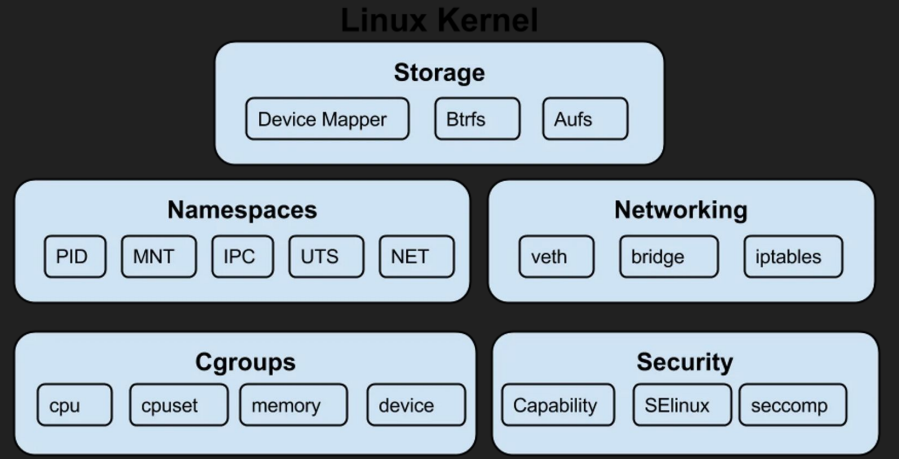
0x02 - Overview of Isolation

Linux Isolation technologies:

- Namespaces
- CGroups
- seccomp-bpf
- Capabilities
- Linux Security Modules (LSM)

Not included in today's workshop:

- ~~Capabilities~~
- ~~Linux Security Modules (LSM)~~
- ~~Filesystems & Storage~~

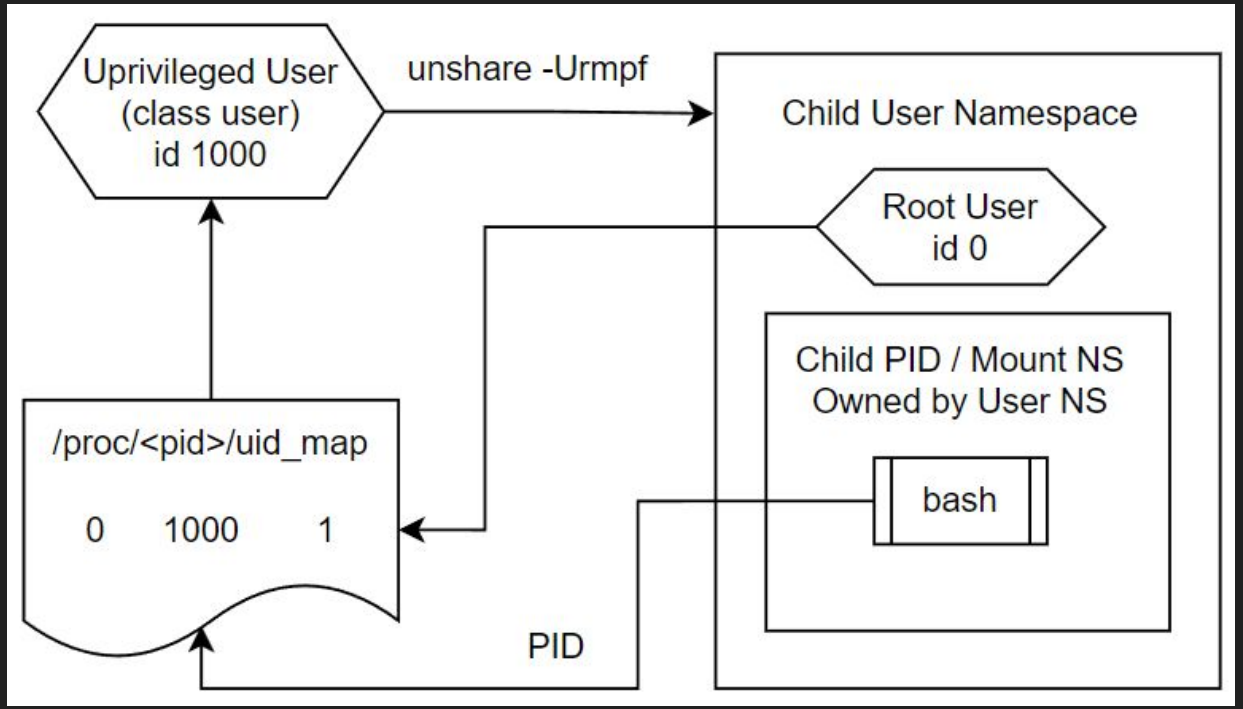


0x03.0 - Linux Namespaces

- Isolates kernel resources
- Kernel-virtualized sandboxes are provided for use by system processes
- The Kernel is shared among all namespaces

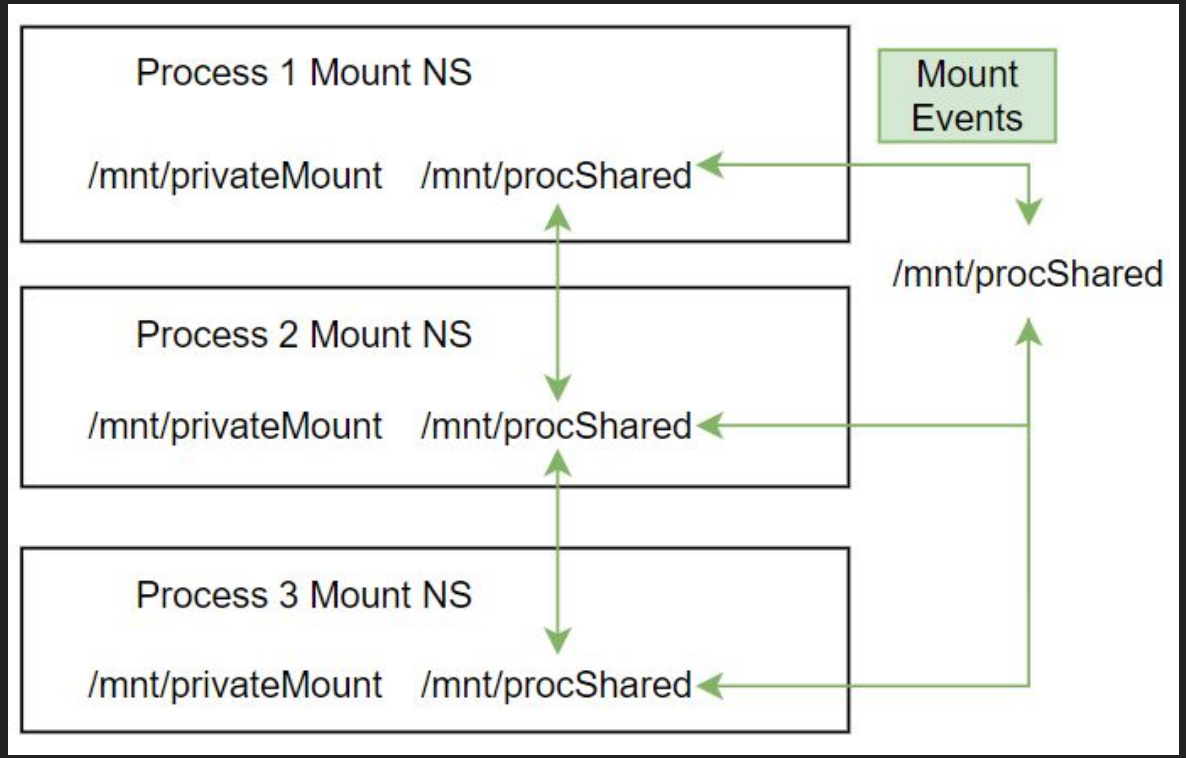
0x03.1 - Linux Namespaces : User

- Isolates:
  - User ids
  - Group ids
  - Keyrings
  - Capabilities
- Maps:
  - userIds
  - groupIds

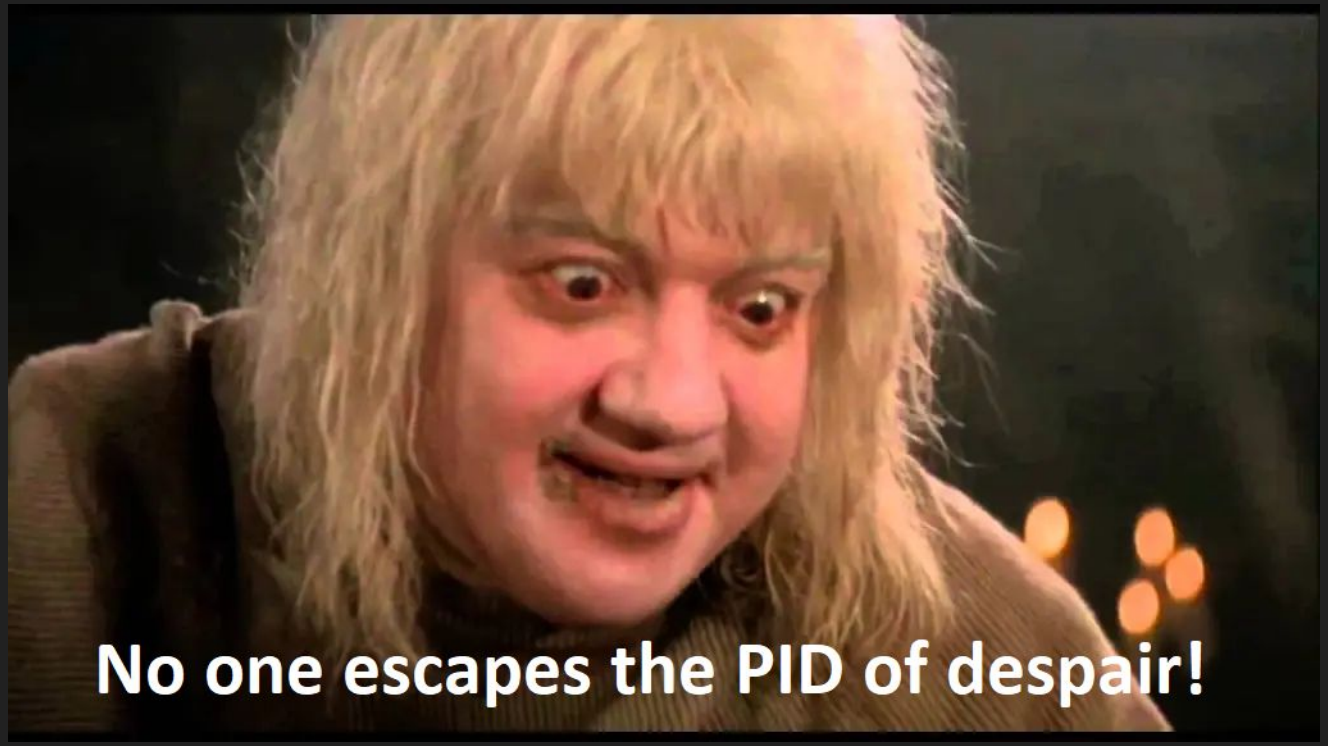


0x03.2 - Linux Namespaces : Mount

- Isolates mount points
- Provides mechanisms to propagate mount events
  - Shared
  - Slave
- ‘Containers’ can isolate their root from host root

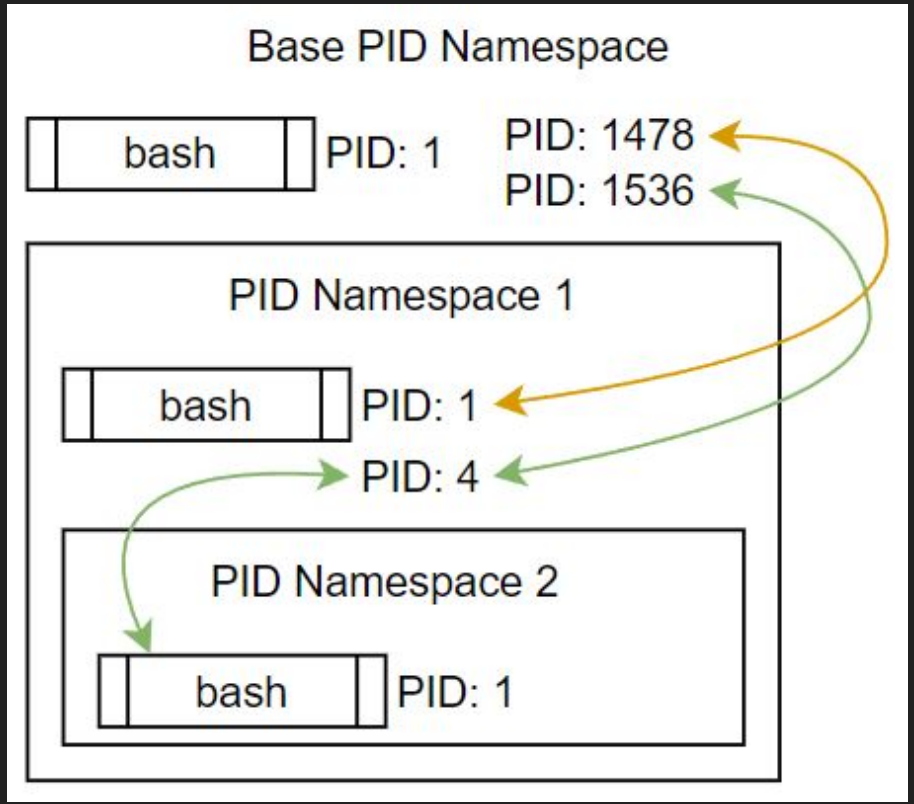


0x03.2 - Linux Namespaces : PID



0x03.2 - Linux Namespaces : PID

- Isolates Process IDs
- Parent PID NS can see processes in child NS
  - But child can’t see in parent
- Used in conjunction with IPC and other namespaces



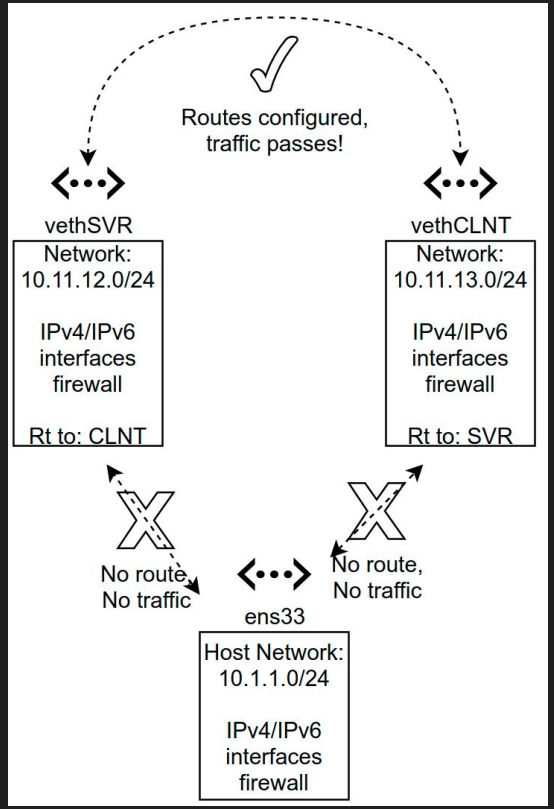
0x03.3 - Linux Namespaces : IPC & Time

- IPC Namespace isolates certain IPC mechanisms
  - SystemV
    - Message Queue
    - Semaphore
    - Shared Memory
  - POSIX Message Queues

- Time Namespace provides offsets for:
  - CLOCK_MONOTONIC
  - CLOCK_BOOTTIME

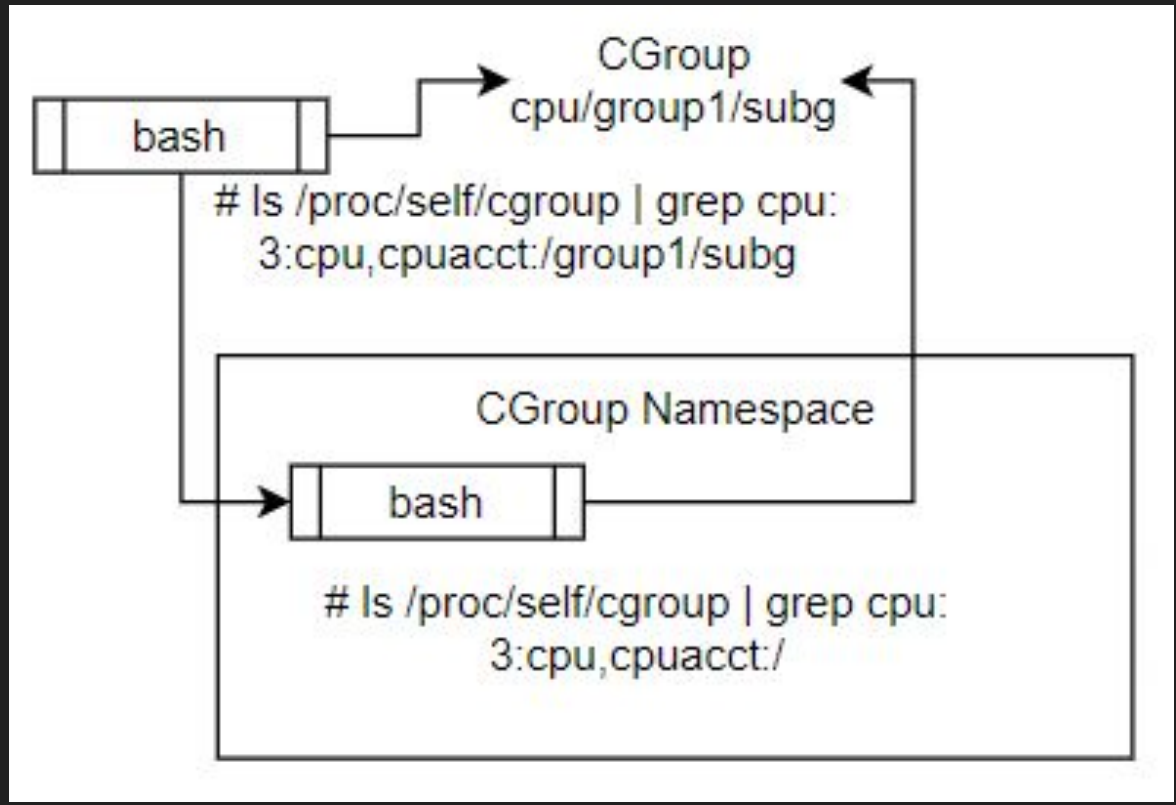
0x03.4 - Linux Namespaces (Network and UTS)

- Isolate ‘global’ resources such as: devices, IP stack, routing & firewall
- Network interfaces can be bound to a single namespace
- Network namespaces are **isolated** from each other
- UTS namespace is a bit useless



0x03.5 - Linux Namespaces : CGroups

- Provides isolation of CGroup naming path
- Allows process migration between systems
- Prevents self-modification of CGroups



0x03.42 - Namespace Labs

- Run the labs

- Also time for questions about the labs

0x04.0 - CGroups (Control Groups)

A CGroup is a set of processes which are bound together under a Subsystem / Resource Controller which limits or controls certain **system resources**.

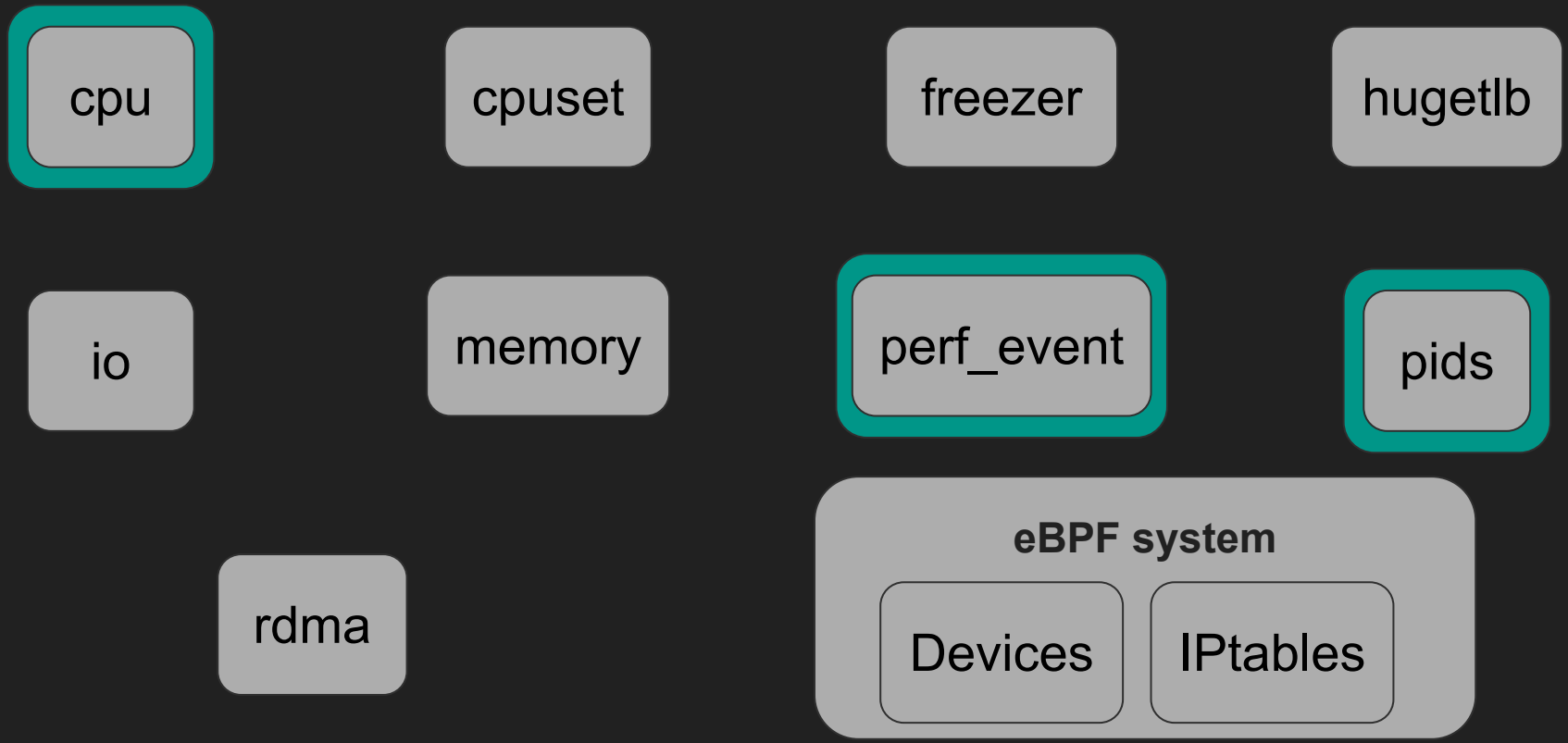
CGroup V1 root on Ubuntu: /sys/fs/cgroup

CGroup V2 root on Ubuntu: /sys/fs/cgroup/unified

Can use V1 or V2 per Subsystem, not both.

0x04.1 - CGroups V1 Subsystems

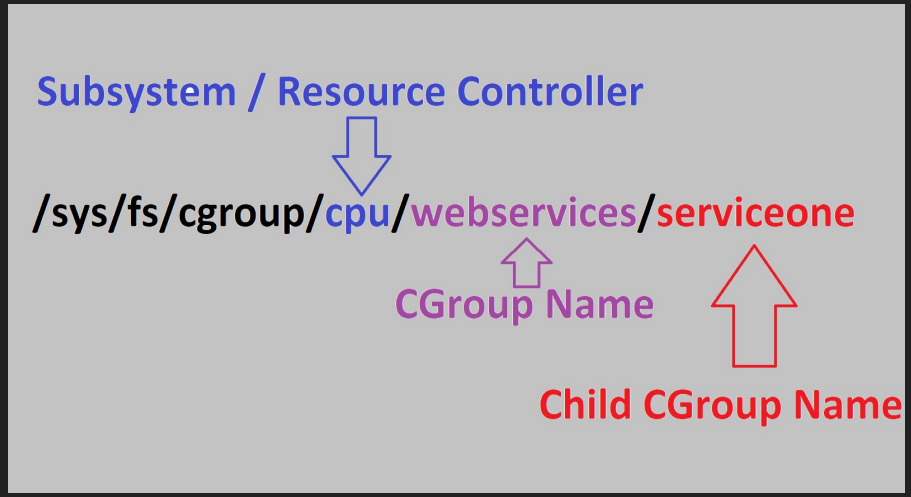
0x04.2 - CGroups V2 Subsystems



0x04.3 - CGroups Hierarchy

CGroup control files are structured in a hierarchy. For example:

- /sys/fs/cgroup/cpu/**webservices**
- /sys/fs/cgroup/cpu/**webservices/serviceone**
- /sys/fs/cgroup/cpu/**webservices/servicetwo**



0x04.4 - CGroups Control File Structure

CGroups each have their own unique control file structure.

Pids CGroup:

- cgroup.clone_children
- cgroup.procs
- notify_on_release
- tasks
- pids.current
- pids.events
- pids.max

Devices CGroup:

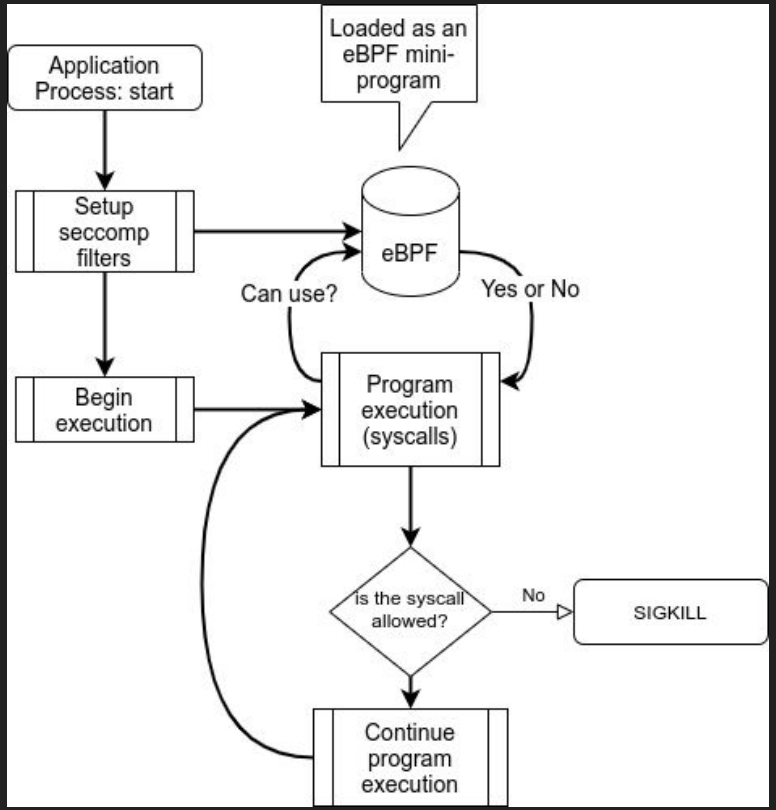
- cgroup.clone_children
- cgroup.procs
- notify_on_release
- tasks
- devices.allow
- devices.deny
- devices.list

0x04.5 - CGroup Labs

- Lab 1, where PIDs are your oyster (until they’re not)
- Lab 2, where CPUs are controlled
- Questions?

0x05 - seccomp

- Restricts the number of syscalls available to a process
- Two modes of operation
  - STRICT
  - FILTER
- FILTER mode is powered by eBPF
  - Whitelist syscalls + arguments
  - Acts as a kernel-mode firewall
- Docker provides a seccomp profile
  - Defaults to block potentially-hazardous syscalls

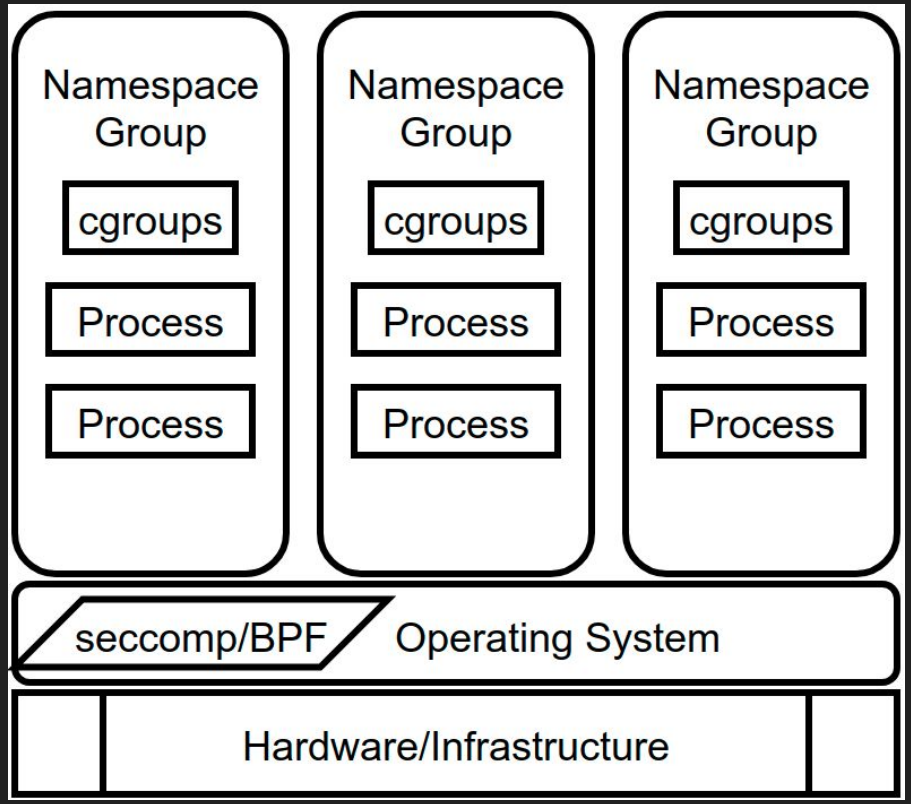


0x05 - seccomp Labs

- Lab 1
- Lab 2
- Questions?

0x06 - In Closing...

- Linux underpins the entire ecosystem
- Each control mechanism has its own knobs to turn
- Taken together these mechanisms provide for light-weight and effective isolation of processes and resources (aka `containerization`)



Questions?

Now is your time!

Thank you!

Rion Carter

- 7thzero.com

Jacob Carter

- androda.work

Further Reading

Always start with
“The Man”

- https://man7.org/linux/man-pages/man7/namespaces.7.html
- https://man7.org/linux/man-pages/man7/cgroups.7.html

- https://www.schutzwerk.com/en/43/posts/linux_container_intro/
- https://news.ycombinator.com/item?id=29265061
- https://redhatgov.io/workshops/containers_the_hard_way/
- https://www.redhat.com/sysadmin/mount-namespaces
- https://linuxcontainers.org/lxd/introduction/
- https://lwn.net/Articles/740157/
- ‘Precursor technology’ like jails, chroot, Solaris stuff, IBM stuff

Image Credits

- Environment Setup slide
  - https://pixabay.com/illustrations/synthwave-retrowave-synth-3941721/
- Image on Introduction slide based on:
  - https://en.wikipedia.org/wiki/Terry%27s_Chocolate_Orange#/media/File:Terrys-Chocolate-Orange.jpg
  - https://commons.wikimedia.org/wiki/User:Evan-Amos
- Thank you slide
  - https://pixabay.com/illustrations/city-sunset-night-landscape-urban-5848267/

Others that are cool:

https://www.flickr.com/photos/xmodulo/26534955924 (namespaces)

https://commons.wikimedia.org/wiki/File:Linux_kernel_map.svg (linux kernel)

https://commons.wikimedia.org/wiki/File:Zombie_process.png (zombie process)