May 17, 2019

Rootless Containers & Unresolved Issues

Akihiro Suda / NTT (@_AkihiroSuda_)

Agenda

- Introduction to Rootless Containers
- How it works
- Adoption status
- Unresolved issues
- containerd dev plan

Introduction

Rootless Containers

- Run containers, runtimes, and orchestrators as a non-root user

- Don’t confuse with:
  - `usermod -aG docker penguin`
  - `docker run --user`
  - `dockerd --userns-remap`

Motivation of Rootless Containers

- To mitigate potential vulnerability of container runtimes and orchestrator (the primary motivation)

- To allow users of shared machines (e.g. HPC) to run containers without the risk of breaking other users environments
  - Still unsuitable for “multi-tenancy” where you can’t really trust other users

- To isolate nested containers, e.g. “Docker-in-Docker”

Runtime vulnerabilities ⚠️

- Docker "Shocker" (2014)
  - A malicious container was allowed to access the host file system, as `CAP_DAC_READ_SEARCH` was effective by default

- Docker CVE-2014-9357
  - A malicious `docker build` container could run arbitrary binary on the host as the root due to an LZMA archive issue

- containerd #2001 (2018)
  - A malicious container image could remove `/tmp` on the host when the image was pulled (not when actually launched!)

Runtime vulnerabilities ⚠️

- Docke
  - A n,
    as

Vulnerability of daemons, not containers per se
So `--userns-remap` is not effective

- Docker CVE-2014-9357
  - A malicious `docker build` container could run arbitrary binary on the host as the root due to an LZMA archive issue

- containerd #2001 (2018)
  - A malicious container image could remove `/tmp` on the host when the image was pulled (not when actually launched!)

Runtime vulnerabilities ⚠️

- runc #1962 (2019)
  - Container break-out via `/proc/sys/kernel/core_pattern` or `/sys/kernel/uevent_helper`
  - Hosts with the initrd rootfs (`DOCKER_RAMDISK`) were affected (e.g. Minikube)

- runc CVE-2019-5736
  - Container break-out via `/proc/self/exe`

Other vulnerabilities ⚠️

- Kubernetes CVE-2017-1002101, CVE-2017-1002102
  - A malicious container was allowed to access the host filesystem via vulnerabilities related to volumes

- Kubernetes CVE-2018-1002105
  - A malicious API call could be used to gain `cluster-admin` (and hence the root privileges on the nodes)

- Git CVE-2018-11235 (affected Kubernetes `gitRepo` volumes)
  - A malicious repo could execute an arbitrary binary as the root when it was cloned

Other vulnerabilities ⚠️

- Kubernetes CVE-2017-1002101, CVE-2017-1002102
  - A malicious container was allowed to access the host filesystem via vulnerabilities related to volumes
- Kubernetes CVE-2018-1002105
  - A ma `in` (and
    henc

`--userns-remap` might not be effective

- Git CVE-2018-11235 (affected Kubernetes `gitRepo` volumes)
  - A malicious repo could execute an arbitrary binary as the root when it was cloned

Play-with-Docker.com vulnerability ⚠️

- Play-with-Docker.com: Online Docker playground, implemented using Docker-in-Docker with custom AppArmor profiles

- Malicious kernel module was loadable due to AppArmor misconfiguration (revealed on Jan 14, 2019)
  - Not really an issue of Docker

https://www.cyberark.com/threat-research-blog/how-i-hacked-play-with-docker-and-remotely-ran-code-on-the-host/

What Rootless Containers can

- Prohibit accessing files owned by other users
- Prohibit modifying firmware and kernel (→ undetectable malware)
- Prohibit other privileged operations like ARP spoofing, rebooting,...

What Rootless Containers cannot

- If a container was broke out, the attacker still might be able to
  - Mine cryptocurrencies
  - Springboard-attack to other hosts

- Not effective for kernel / VM/ HW vulns
  - But we could use gVisor together for mitigating some of them

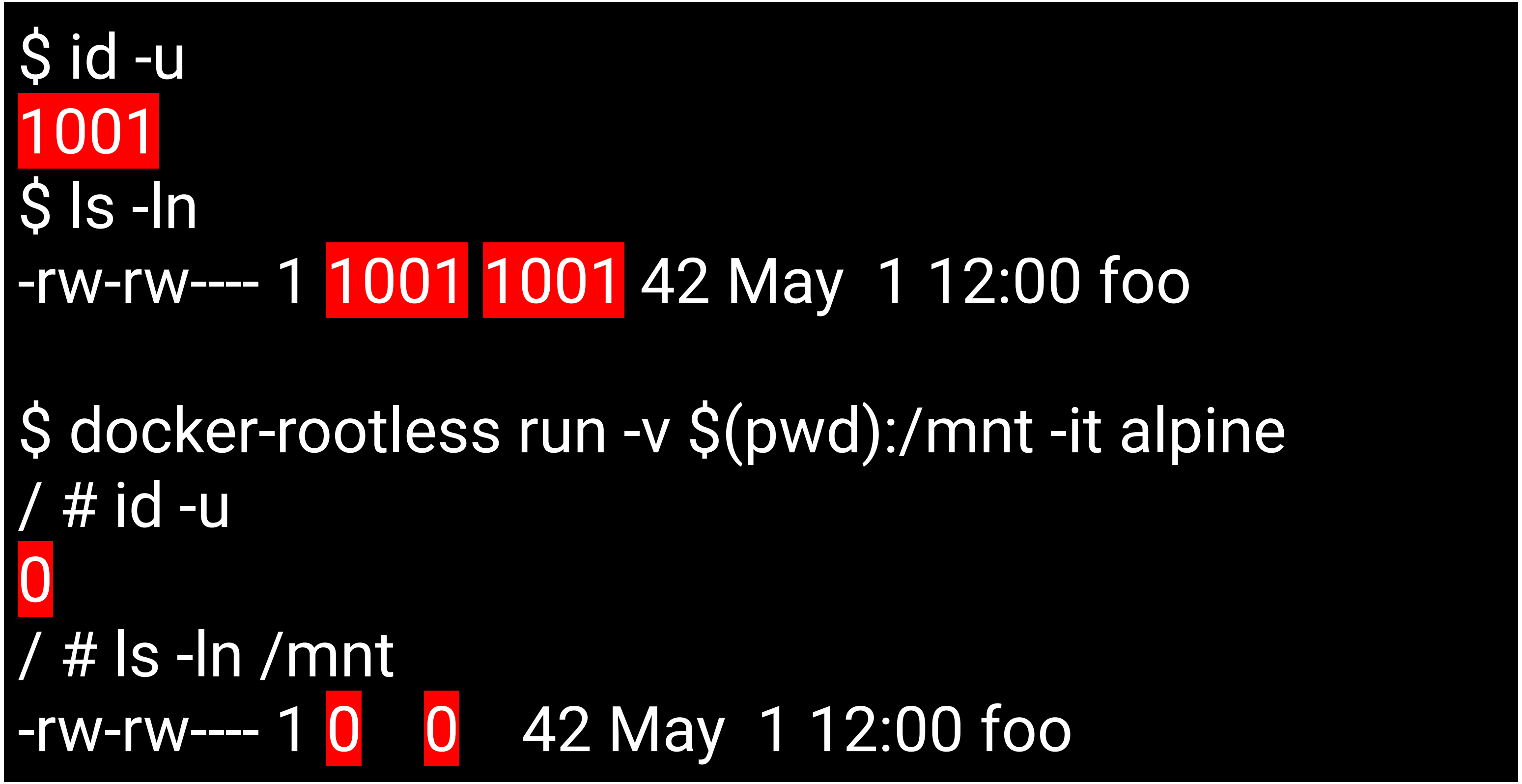
How it works

User Namespaces

- User namespaces allow non-root users to pretend to be the root

- Root-in-UserNS can have “fake” UID 0 and also create other namespaces (MountNS, NetNS..)

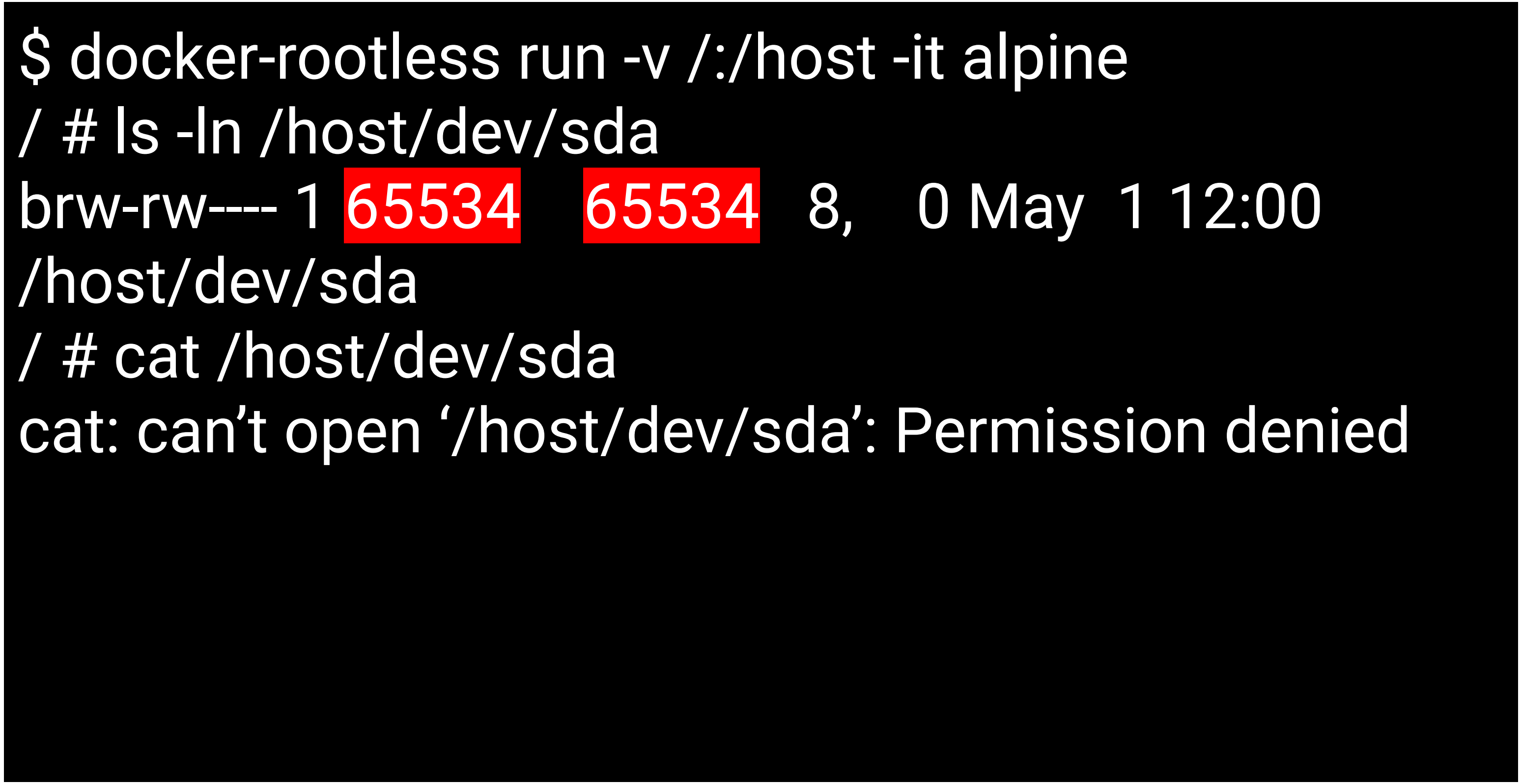
User Namespaces



```
$ id -u
1001
$ ls -ln
-rw-rw---- 1 1001 1001 42 May  1 12:00 foo

$ docker-rootless run -v $(pwd):/mnt -it alpine
/ # id -u
0
/ # ls -ln /mnt
-rw-rw---- 1 0    0    42 May  1 12:00 foo
```

User Namespaces



```
$ docker-rootless run -v /:/host -it alpine
/ # ls -ln /host/dev/sda
brw-rw---- 1 65534    65534   8,    0 May  1 12:00
/host/dev/sda
/ # cat /host/dev/sda
cat: can’t open ‘/host/dev/sda’: Permission denied
```

Sub-users (and sub-groups)

- Put users in your user account so you can be a user while you are being a user

- Sub-users are used as non-root users in a container
  - `USER` in Dockerfile
  - `docker run --user`

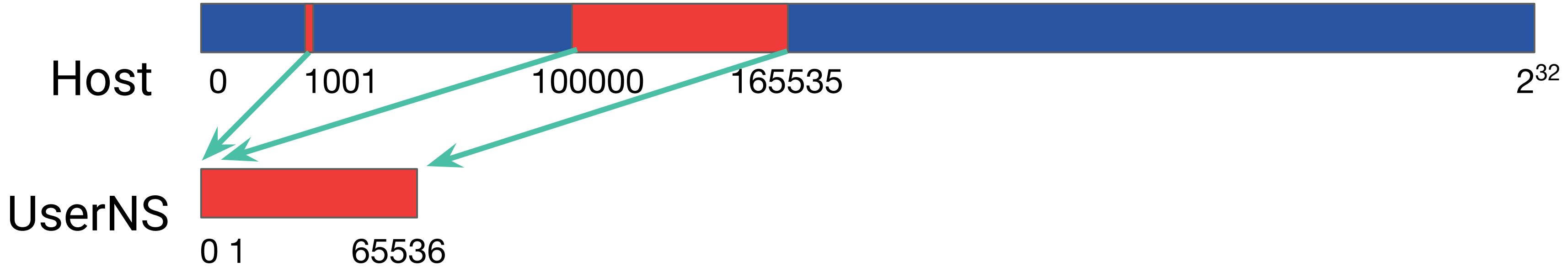
Sub-users (and sub-groups)

- If `/etc/subuid` contains “`1001:100000:65536`”

primary user

sub-users start

sub-users length



- Having 65536 sub-users should be enough for most containers

Sub-users (and sub-groups)

- Sub-users are configured via SUID binaries `/usr/bin/{newuidmap, newgidmap}`

- SETUID binary can be dangerous; `newuidmap` & `newgidmap` had two CVEs so far:
  - CVE-2016-6252 (CVSS v3: 7.8): integer overflow issue
  - CVE-2018-7169 (CVSS v3: 5.3): supplementary GID issue

Sub-users (and sub-groups)

- Also hard to maintain sub-users
  - LDAP / AD
  - Nesting user namespaces might need huge number of sub-users

Sub-users (and sub-groups)

- Alternative way: Single-mapping mode
- Does not require `newuidmap/newgidmap`
- Ptrace and/or Seccomp can be used for intercepting syscalls to emulate sub-users
  - `user.rootlesscontainers` xattr can be used for chown emulation

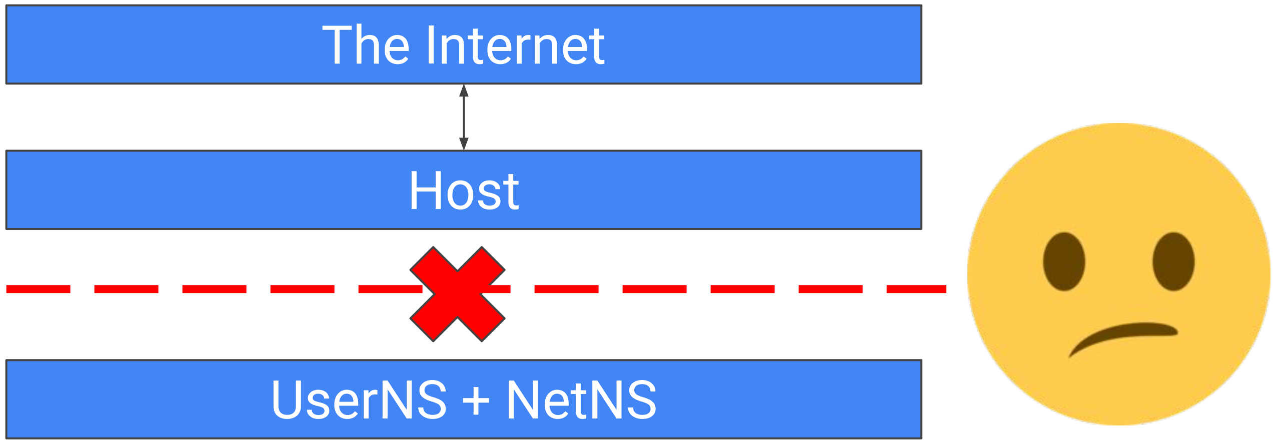
Network Namespaces

- An unprivileged user can create network namespaces along with user namespaces

- With network namespaces, the user can
  - isolate abstract (pathless) UNIX sockets
    - important to prevent container breakout
  - create iptables rules
  - set up overlay networking with VXLAN
  - run tcpdump
  - ...

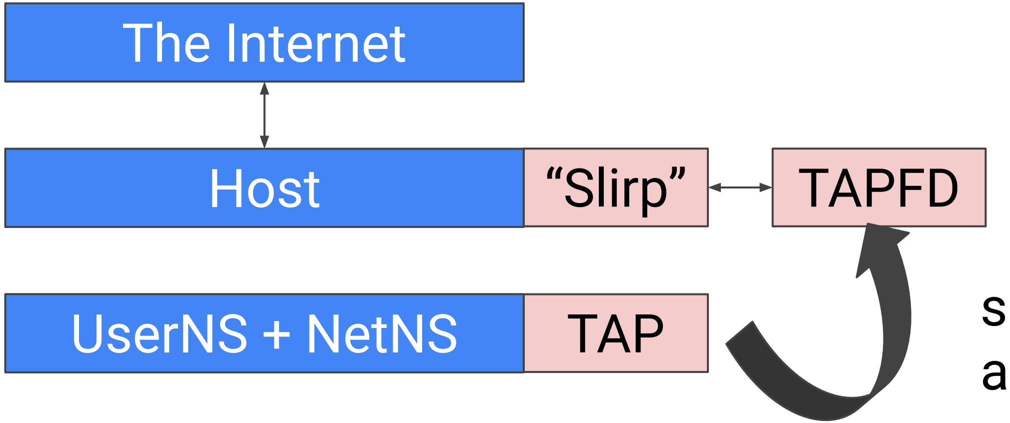
Network Namespaces

- But an unprivileged user cannot set up `veth` pairs across the host and namespaces, i.e. No internet connection



Network Namespaces

- `lxc-user-nic` SUID binary allows unprivileged users to create veth, but we are not huge fun of SUID binaries
- Our approach: use completely unprivileged usermode network ("Slirp") with a TAP device



send fd as
a `SCM_RIGHTS` cmsg

Network Namespaces

Benchmark of several “Slirp” implementations:

	MTU=1500	MTU=4000	MTU=16384	MTU=65520
vde_plug	763 Mbps	Unsupported	Unsupported	Unsupported
VPNKit	514 Mbps	526 Mbps	540 Mbps	Unsupported
slirp4netns	1.07 Gbps	2.78 Gbps	4.55 Gbps	9.21 Gbps
cf. rootful veth	52.1 Gbps	45.4 Gbps	43.6 Gbps	51.5 Gbps

- slirp4netns (our own implementation based on QEMU Slirp) is the fastest because it avoids copying packets across the namespaces

Benchmark: iperf3 (netns -> host), measured on Travis CI. See rootless-containers/rootlesskit#12

Multi-node networking

- Flannel VXLAN is known to work
  - Encapsulates Ethernet packets in UDP packets
  - Provides L2 connectivity across rootless containers on different nodes

- Other protocols should work as well, except ones that require access to raw Ethernet

Snapshotting

- OverlayFS is currently unavailable in UserNS (except on Ubuntu kernel)

- FUSE-OverlayFS can be used instead with kernel 4.18+

- XFS reflink can be also used to deduplicate files (but slow)

Cgroup

- `pam_cgfs` can be used for delegating permissions to unprivileged users, but considered insecure by systemd folks https://github.com/containers/libpod/issues/1429

- cgroup2 provides proper support for delegation, but not adopted by OCI at the moment

Rootless Containers in Containers

- Urge demand for building images on Kubernetes cluster
- Seccomp and AppArmor needs to be disabled for the parent containers
- To allow the children to mount procfs (pid-namespaced), `maskedPaths` and `readonlyPaths` for `/proc/*` for the parent needs to be removed (weird!)
  - Same applies to sysfs (net-namespaced)

Rootless Containers in Containers

- So `--privileged` had been typically required anyway :(
  - Or at least `--security-opt {seccomp,apparmor}=unconfined`

- Docker 19.03 supports `--security-opt systempaths=unconfined` for allowing procfs & sysfs mount (Kube: `securityContext.procMount`, but no `sysMount` yet)
  - Make sure to lock the root in the container! (`passwd -l root`, Alpine CVE-2019-5021)

Adoption status

Adoption status: runtimes

	Docker v19.03 containerd runc	**Podman (≈ CRI-O) crun**	**LXC**	**Singularity**
NetNS isolation with Internet connectivity	● VPNKit ● slirp4netns ● lxc-user-nic (SUID)	slirp4netns	lxc-user-nic (SUID)	No support
Supports FUSE-OverlayFS	No	Yes	No	No
Cgroup	No	Limited support for cgroup2	pam_cgfs	No

Adoption status: runtimes::GPU

- nvidia-container-runtime is known to work
- Need to disable cgroup manually
- Rootful nVIDIA container needs to be executed on every system startup
- Probably, other devices such as FPGA should work as well (untested)

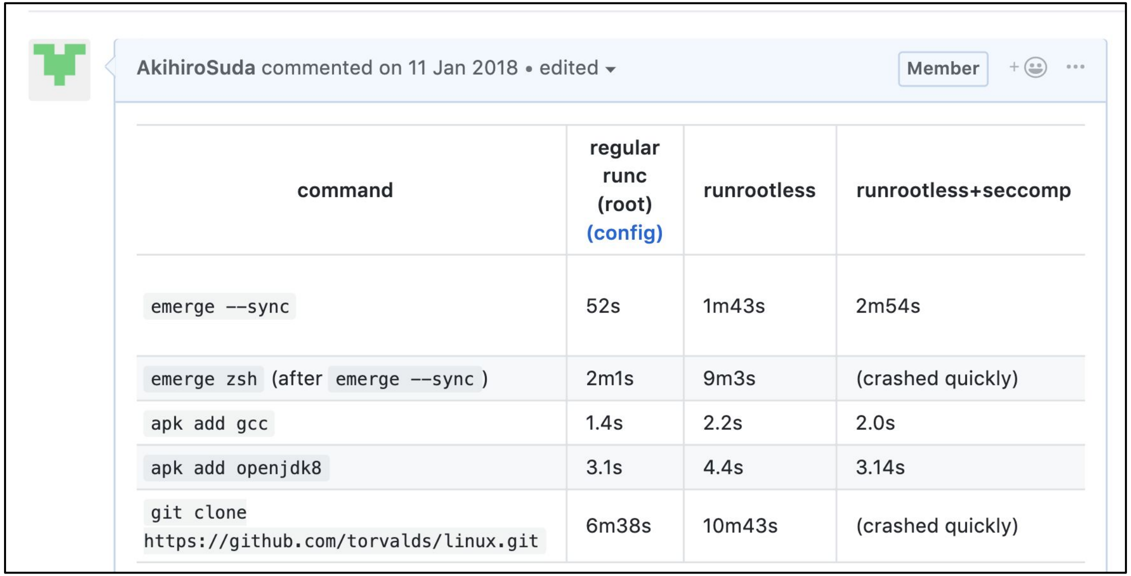
Adoption status: runtimes::single-mapping mode

- udocker does not need subuid configuration, as it can emulate subuser with ptrace (based on PRoot)
  - but no persistent `chown`

- runROOTLESS (Don't confuse with upstream rootless runc) supports persistent `chown` as well, using `user.rootlesscontainers` xattr
  - the xattr value is a pair of UID and GID in protobuf encoding
  - the xattr convention is compatible with umoci

Adoption status: runtimes::single-mapping mode

- Ptrace is slow https://github.com/rootless-containers/runrootless/issues/14
- seccomp can be used for acceleration but hard to implement correctly



AkihiroSuda commented on 11 Jan 2018 • edited

command	regular runc (root) (config)	runrootless	runrootless+seccomp
`emerge --sync`	52s	1m43s	2m54s
`emerge zsh` (after `emerge --sync`)	2m1s	9m3s	(crashed quickly)
`apk add gcc`	1.4s	2.2s	2.0s
`apk add openjdk8`	3.1s	4.4s	3.14s
`git clone https://github.com/torvalds/linux.git`	6m38s	10m43s	(crashed quickly)

Adoption status: image builders

- BuildKit / img / Buildah supports rootless mode
  - Works in containers as well as on the host
  - Does not need `--privileged` but Seccomp and AppArmor needs to be disabled

Adoption status: image builders

- Similar but different work: Kaniko & Makisu
  - Rootful
  - But no need to disable seccomp and AppArmor, because they don’t create containers for RUN instructions in Dockerfile

Adoption status: Kubernetes

- Usernetes project provides patches for rootless Kubernetes, but not proposed to the upstream yet
  - Supports all major CRI runtimes: dockershim, containerd, CRI-O
  - Flannel VXLAN is known to work
  - Lack of cgroup might be huge concern

- But Usernetes is already integrated into k3s!
  (*5 less than k8s*)

```
$ k3s server --rootless
```

You can *rootlesify* your own project easily!

- RootlessKit does almost all things for *rootlessifying* your container project (or almost any rootful app)
  - Creates UserNS with sub-users and sub-groups
  - Creates MountNS with writable /etc, /run but without chroot
  - Creates NetNS with VPNKit/slirp4netns/lxc-user-nic
  - Provides REST API on UNIX socket for port forwarding management

You can *rootlesify* your own project easily!

```
$ rootlesskit --net=slirp4netns --copy-up=/etc  \
  --port-driver=builtin bash
# id -u
0
# touch /etc/here-is-writable-tmpfs
# ip a
...
2: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP>
    inet 10.0.2.100/24 scope global tap0
...
# rootlessctl add-ports 0.0.0.0:8080:80/tcp
```

You can *rootlesify* your own project easily!

- With RootlessKit, you just need to work on disabling cgroup stuff, sysctl stuff, and changing the data path from /var/lib to /home

- Used by Docker, BuildKit, k3s

Unresolved Issues

Kernel has vulns

- UserNS tends to have priv escalation vulns
  - CVE 2013-1858: UserNS + CLONE_FS
  - CVE-2014-4014: UserNS + chmod
  - CVE-2015-1328: UserNS + OverlayFS (Ubuntu-only)
    - So rootless OverlayFS is still not merged in upstream
  - CVE-2018-18955: UserNS + complex ID mapping

Kernel has vulns

- A bunch of code paths that can hang up the kernel
  - e.g. CVE-2018-7191 (~~*unpublished*~~ *published today*): creating a tap device with illegal name
  - And more, see https://medium.com/@jain.sm/security-challenges-with-kubernetes-818fad4a89f2

- Unlimited resources e.g.
  - Pending signals
  - Max user process
  - Max FDs per user

    (see the same URL above)

Kernel has vulns

- So I’ve never suggested using rootless containers for real multi-tenancy ¯_(ツ)_/¯

Kernel has vulns

- gVisor might be able to mitigate them but significant overhead and syscall incompatibility

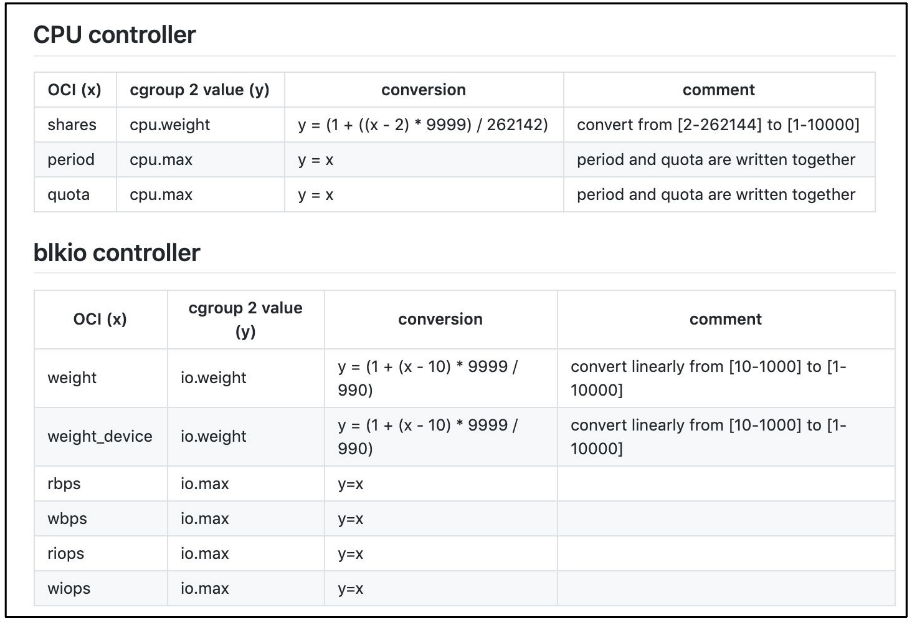
- UML (20 yo, still alive!) is almost compatible with real Linux but it even lacks support for SMP

- linuxd: similar to UML but accelerated with host kernel patches
  - Still no public code

  https://schd.ws/hosted_files/ossna18/db/Containerize%20Linux%20Kernel.pdf

Cgroups

- cgroup2 is not adopted in OCI
- crun is trying to support cgroup2 without changing OCI spec



CPU controller

OCI (x)	cgroup 2 value (y)	conversion	comment
shares	cpu.weight	y = (1 + ((x - 2) * 9999) / 262142)	convert from [2-262144] to [1-10000]
period	cpu.max	y = x	period and quota are written together
quota	cpu.max	y = x	period and quota are written together

blkio controller

OCI (x)	cgroup 2 value (y)	conversion	comment
weight	io.weight	y = (1 + (x - 10) * 9999 / 990)	convert linearly from [10-1000] to [1-10000]
weight_device	io.weight	y = (1 + (x - 10) * 9999 / 990)	convert linearly from [10-1000] to [1-10000]
rbps	io.max	y=x	
wbps	io.max	y=x	
riops	io.max	y=x	
wiops	io.max	y=x	

Mount

- Only supports:
  - tmpfs
  - bind
  - procfs (PID-namespaced)
  - sysfs (net-namespaced)
  - FUSE (since kernel 4.18)
  - Overlay (Ubuntu only)

- No support for mounting any block devices (even loopback devices)

Landlock

- landlock: unprivileged sandbox LSM

- Not merged in the upstream kernel, but promising as AppArmor-alternative

LDAP / Active Directory

- `/etc/sub{u,g}id` configuration is painful for LDAP/AD

- Alternatively, implementing NSS module is under discussion, but no code yet https://github.com/shadow-maint/shadow/issues/154

Single-mapping mode

- runROOTLESS / PRoot could be accelerated with seccomp but implementation is broken

- Kernel 5.0 seccomp could be used for getting rid of ptrace completely

containerd dev plan

containerd dev plan

- Implement FUSE-OverlayFS snapshotter plugin
  - Probably in a separate repo
  - Should not be difficult

- Support cgroup2
  - Probably we want to wait for OCI Runtime Spec and runc to be revised
  - But we can also consider beginning support cgroup2 right now with crun

containerd dev plan

- Support running containerd inside gVisor
  - So as to allow running rootless containers in a container without disabling seccomp & apparmor
  - And to mitigate potential kernel vulns
  - Currently MountNS is not working
    https://github.com/google/gvisor/issues/221