FOSDEM (Feb 3, 2019)

Rootless Kubernetes

Running Kubernetes and CRI/OCI Runtimes as an unprivileged user

Akihiro Suda / NTT (@_AkihiroSuda_)

Giuseppe Scrivano / Red Hat (@gscrivano)

Who are we?

Akihiro Suda

- Software Engineer at NTT (the largest telco in Japan)
- Maintainer of Moby (former Docker Engine), BuildKit, containerd, and etc...

Giuseppe Scrivano

- Software Engineer at Red Hat
- Works mostly on Podman, Buildah, CRI-O

Demo

Introduction

Rootless Kubernetes

- Not just about running containers as an unprivileged user
  - `securityContext.runAsUser`
  - Node-level UserNS (Kubernetes 1.14)

- We run everything as an unprivileged user!
  - OCI Runtimes
  - CRI Runtimes
  - kubelet, kube-proxy, kube-apiserver, kube-scheduler...

Motivation of Rootless Kubernetes

- To mitigate potential vulnerability of OCI/CRI runtimes and Kubernetes itself (the primary motivation)
- To allow users of shared machines (e.g. HPC) to run Kubernetes without the risk of breaking other users environments
- Kubernetes-on-Kubernetes

Kubernetes vulnerabilities

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

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

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

Minikube breakout

- runc #1962 (2019, found by Akihiro, analyzed and fixed by Giuseppe)
  - A malicious container could gain the write access to `/proc` and `/sys` when the host root filesystem is initrd (`DOCKER_RAMDISK`)
    - Results in arbitrary command execution as the root on the host, via `/proc/sys/kernel/core_pattern` or `/sys/kernel/uevent_helper`
  - Minikube is known to be affected (fixed in v0.33.1)

```
$ kubectl run -it --image busybox foo
# unshare -mrfp
# mount -t proc none /proc
```

How it works

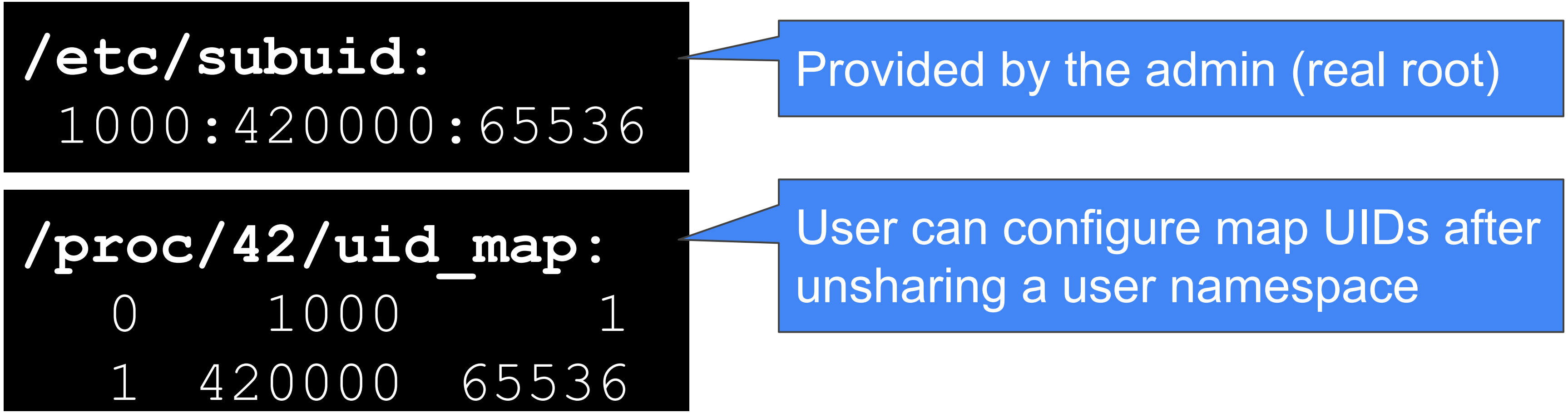
User Namespaces

- The key component of rootless containers.
  - Map UIDs/GIDs in the guest to different UIDs/GIDs on the host.
  - Unprivileged users can have (limited) root inside a user namespace! 👀

- Root in a user namespace has UID 0 and full capabilities, but obvious restrictions apply.
  - Inaccessible files, inserting kernel modules, rebooting, ...

User Namespaces

- To allow multi-user mappings, shadow-utils provides `newuidmap` and `newgidmap` (packaged by most distributions).
  - SETUID binaries writing mappings configured in `/etc/sub[ug]id`



```
/etc/subuid:
 1000:420000:65536
```

Provided by the admin (real root)

```
/proc/42/uid_map:
   0     1000       1
   1   420000   65536
```

User can configure map UIDs after unsharing a user namespace

Network Namespaces

- An unprivileged user can create network namespaces along with user namespaces
  - For iptables, VXLAN, abstract socket isolation…

- But an unprivileged user cannot set up `veth` pairs across the host and namespaces, i.e. No internet connection
  - User-mode network stack (“Slirp”) can be used instead

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

- 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

Root Filesystems

Your container root filesystem has to live *somewhere*. Many filesystem features used by “rootful” container runtimes aren’t available.

- Ubuntu allows overlayfs in a user namespace, but this isn't supported upstream (due to security concerns).
- Btrfs allows unprivileged subvolume management, but requires privileges to set it up beforehand.
- Devicemapper is completely locked away from us.

Root Filesystems

A “simple” work-around is to just extract images to a directory!

- It works … but people want storage deduplication.

Alternatives:

- Reflinks to a "known good" extracted image (inode exhaustion).
  - (Can use on XFS, btrfs, ... but not ext4.)
- Unprivileged userspace overlayfs using FUSE (Kernel 4.18+).

fuse-overlayfs

- Overlayfs implementation using FUSE
- Layers deduplication as for root containers
- Fast setup for a new container

- Adds complexity
- Temporary solution until unprivileged users can safely use overlay

cgroups

- cgroups v1 delegation to unprivileged users is not safe

- cgroups v2 supports delegation to unprivileged users, but v2 is not adopted in the current OCI ecosystem yet

Implementation in Kubernetes

Implementation in Kubernetes

- `kubelet` and `kube-proxy` need to be patched
  - cgroups and some of sysctl need to be disabled
  - Our patches will be proposed to SIG-node soon

- CRI: Both CRI-O and containerd supports rootless mode
  - Docker v19.03 is likely to support rootless mode

- CNI: Flannel VXLAN is known to work without any modification

- `kubeadm` integration is on plan

“Usernetes”

Experimental binary distribution of rootless Kubernetes, installable under `$HOME` without mess

https://github.com/rootless-containers/usernetes

```
$ tar xjvf usernetes-x86_64.tbz
$ cd usernetes
$ ./run.sh
```

```
$ ./kubectl.sh run -it --image..
```

“Usernetes”

- `docker-compose.yml` is included for demonstrating pseudo multi-node cluster POC
  - Mix of dockershim + CRI-O + containerd
  - Flannel VXLAN is enabled by default
  - FIXME: TLS is not enabled yet 😝 (contribution wanted!)

- Usernetes-on-Kubernetes YAML is coming soon

Any questions?