Rootless Containers 2020

Akihiro Suda (containerd / NTT)

containerd

Rootless Containers 2020

Akihiro Suda (containerd / NTT)

Ask me questions at
#2-kubecon-maintainer (https://slack.cncf.io)

containerd

What is Rootless Containers?

- Running container runtimes (and also containers, of course) as a non-root user on the host
  - OCI (e.g. runc)
  - CRI (e.g. containerd)
  - CNI (e.g. Flannel)
  - kubelet, dockerd, …
- Protects the host from potential vulnerabilities and misconfigurations

What is Rootless Containers?

Don’t be confused… The following stuffs are unrelated:

- `.spec.securityContext.runAsUser` (≈ `docker run --user`)
- UserNS KEP (≈ `dockerd --userns-remap`)
- `usermod -aG docker foo`
- Singularity with SETUID

Why do we need Rootless?

Most runtimes are designed to be secure by default, but they are still likely to have vulnerabilities

Identifier	Component	Description
CVE-2017-1002102	kubelet	Files on the host could be removed
containerd#2001 (2018)	containerd	/tmp on the host could be removed
CVE-2018-11235	kubelet	Arbitrary command could be executed on the host
runc#1962 (2019)	runc	Bare procfs was exposed with non-pivot rootfs mode
CVE-2019-5736	runc	runc binary could be replaced with a malicious file
CVE-2019-11245	kubelet	An image could be executed with an unexpected UID
CVE-2019-14271	dockerd	A malicious NSS library could be loaded
…	…	…
And more! 		

Why do we need Rootless?

- People often make misconfigurations ☹
  - Sets up insufficient PodSecurityPolicy / Gatekeeper policies
  - Exposes system components' TCP ports without mTLS

    (e.g. etcd, kube-apiserver, kubelet, dockerd…)
  - Exposes private keys as IaaS metadata (169.254.169.254)
  - Uses same kubelet certs for all the nodes
  - …

Why do we need Rootless?

- Rootless Containers can mitigate the impacts of such vulnerabilities and misconfiguration

- Even if the host gets compromised, the attacker won't be able to:
  - access files owned by other users
  - modify firmware and kernel (→ undetectable malware)
  - ARP spoofing (→ DNS spoofing)

Not a panacea, of course…

Not effective against:

- Vulnerabilities of kernel and hardware
- DDoS attacks
- Cryptomining …

Not a panacea, of course…

Some caveats apply

- Network throughput is slowed down

  (But we are seeing HUGE improvements in 2020)

- No support for NFS and block storages

  (But it doesn't matter if you use managed DBs and object storages)

History

It began in c. 2012… But wasn't popular until 2018-2019

Year	Low layers	High layers
2012	Kernel [officially in 2013]	
2013	Semi-privileged networking with SETUID	LXC
2014		
2015		
2016		runc [officially in 2017]
2017		

History

It began in c. 2012… But wasn't popular until 2018-2019

Year	Low layers	High layers
2018	Unprivileged networking (slirp4netns) Unprivileged FUSE-OverlayFS	BuildKit, based on containerd tech Docker [officially in 2019] & containerd Podman & CRI-O Kubernetes [unofficial, still]
2019	Unprivileged cgroup v2 via systemd Faster port forwarding (RootlessKit)	k3s
2020	Faster networking with seccomp addfd	
2021+		Kubernetes, officially?

Example: Docker

- https://get.docker.com/rootless
- Rootless mode was experimental in v19.03, will be GA in v20.10
- Other notables updates in v20.10 w.r.t. Rootless:
  - Resource limitation with Cgroup v2
  - FUSE-OverlayFS
  - Improved installer

Example: Docker

Easy to install

```
$ curl -fsSL https://get.docker.com/rootless | sh ⏎

$ export DOCKER_HOST=unix:///run/user/1000/docker.sock ⏎

$ docker run -d --name caddy -p 8080:80 caddy ⏎

$ curl http://localhost:8080 ⏎
...
<title>Caddy works!</title>
...
```

Example: Docker

All processes are running as a non-root user

```
$ pstree user ⏎
sshd──bash──pstree

systemd─┬─(sd-pam)
        ├─containerd-shim─┬─caddy──7*[{caddy}]
        │                 └─12*[{containerd-shim}]
        └─rootlesskit─┬─exe─┬─dockerd─┬─containerd──10*[{containerd}]
                      │     │         ├─rootlesskit-doc─┬─docker-proxy──6*[{docker-proxy}]
                      │     │         │                 └─6*[{rootlesskit-doc}]
                      │     │         └─11*[{dockerd}]
                      │     └─11*[{exe}]
                      ├─vpnkit──4*[{vpnkit}]
                      └─8*[{rootlesskit}]
```

Example: Usernetes

- https://github.com/rootless-containers/usernetes
- Rootless Kubernetes distribution
- Multi-node demo is provided as a Docker Compose stack
- CNI: Flannel (VXLAN)

```
$ docker-compose up -d ⏎

$ kubectl get nodes ⏎
NAME                STATUS   ROLES    AGE     VERSION
node-containerd     Ready    <none>   3m46s   v1.19.0-usernetes
node-crio           Ready    <none>   3m46s   v1.19.0-usernetes
```

Example: Usernetes


```
$ docker exec usernetes_node-containerd_1 pstree user ⏎
journalctl---(sd-pam)

systemd-+-(sd-pam)
        |-containerd-fuse---containerd-fuse---4*[{containerd-fuse}]
        |-containerd.sh---containerd---10*[{containerd}]
        |-flanneld.sh---flanneld---9*[{flanneld}]
        |-nsenter.sh---kubelet---13*[{kubelet}]
        |-nsenter.sh---kube-proxy---7*[{kube-proxy}]
        `-rootlesskit.sh---rootlesskit-+-exe-+-rootlesskit.sh---sleep
                                       |     `-9*[{exe}]
                                       |-slirp4netns
                                       `-8*[{rootlesskit}]
```

Example: k3s

- https://k3s.io/
- CNCF Sandbox Project
- Focuses on edge computing
- Incorporates Usernetes patches for supporting rootless, ahead of the Kubernetes upstream
- Uses containerd as the CRI runtime

```
$ k3s server --rootless ⏎

$ k3s kubectl apply -f manifest.yaml ⏎
```

Example: BuildKit

- https://github.com/moby/buildkit
- A container image builder, built on containerd technology
- Can be executed in several ways
  - As a built-in feature of dockerd
  - As a standalone daemon
  - As a Kubernetes Pod
  - As a Kubernetes Job, without a daemon Pod
  - As a Tekton Task

Example: BuildKit

No need to set `securityContext.Privileged`

But Seccomp and AppArmor constraints need to be relaxed

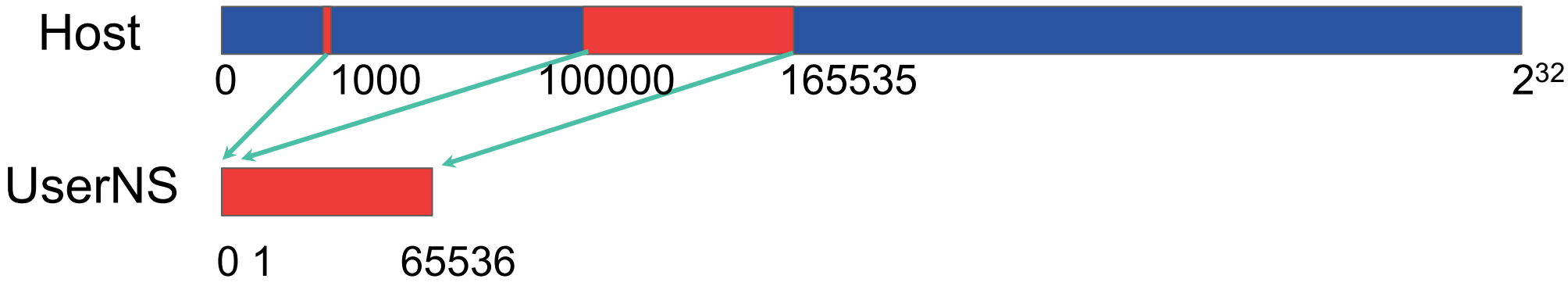
```
spec:
 containers:
  - securityContext:
      runAsUser: 1000
      seccompProfile:
        type: Unconfined

metadata:
  annotations:
    container.apparmor.security.beta.kubernetes.io/buildkitd: unconfined
```

How it works

- UserNS
- MountNS
- NetNS
- Cgroup
- New frontier: Seccomp User Notification

How it works: UserNS

- Maps a non-root user (e.g. UID 1000) to a fake root user (UID 0)
- Not the real root, but enough to run containers
- Subordinate UIDs are mapped as well
  (typically 65,536 UIDs, defined in `/etc/subuid`)

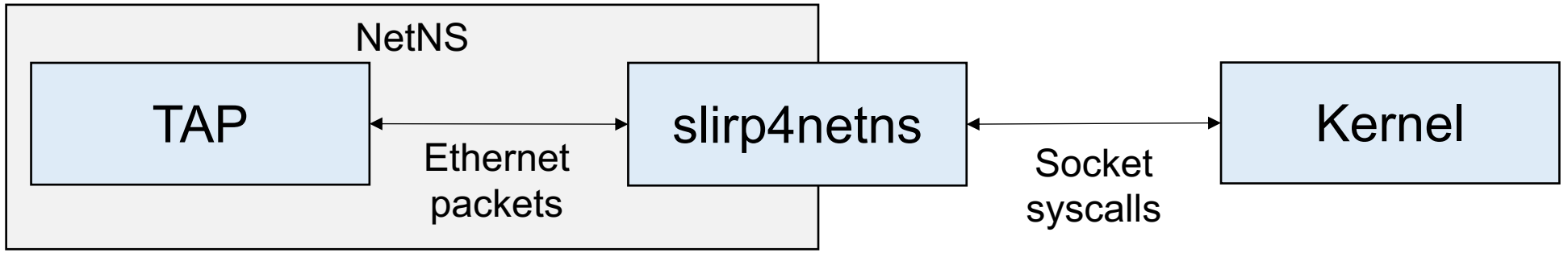


How it works: MountNS

- A non-root user can create MontNS along with UserNS
- But cannot mount most filesystems, except bind-mount, tmpfs, procfs, and sysfs...
  - No Overlayfs (on vanilla kernel)
  - No NFS
  - No block storages
- FUSE is supported since kernel 4.18
- FUSE-OverlayFS can substitute real OverlayFS

How it works: NetNS

- A non-root user can also create NetNS with UserNS
- But cannot create vEth pairs, i.e. No internet connectivity
- Slirp is used instead of vEth for unprivileged internet connectivity



- Slow (51.5Gbps → 9.21Gbps), but we are seeing huge improvements

How it works: Cgroup

- No support for cgroup v1
- i.e. no memory limit, no CPU limit, no fork-bomb guard...
- Cgroup v2 is almost fully supported
- Fedora has already switched the default to v2
- Other distros will follow in 2021-2022 ?

A new frontier in 2020: Seccomp User Notification

- Kernel 5.0 merged the support for Seccomp User Notification: a new way to hook syscalls in the userspace
- Similar to ptrace, but less numbers of context switches
- Allows emulating subordinate UIDs without `/etc/subuid`
- POC: https://github.com/rootless-containers/subuidless

A new frontier in 2020: Seccomp User Notification

- Kernel 5.9 merged the support for `SECCOMP_IOCTL_NOTIF_ADDFD`
- Allows injecting file descriptors from a host process into container processes
- e.g. replace `sockfd` on `connect(2)`
- No slirp overhead any more
- POC: https://github.com/rootless-containers/bypass4netns

Recap

- Rootless Containers can protect the host from potential vulnerabilities and misconfigurations
- Already adopted by lots of projects: BuildKit, Docker, containerd, Podman, CRI-O, k3s ...
- Being also proposed to the Kubernetes upstream
- There are some drawbacks, but being significantly improved using Seccomp User Notification

Resources

- Rootless Containers overview: https://rootlesscontaine.rs/
- Rootless containerd: https://github.com/containerd/containerd/blob/master/docs/rootless.md
- Rootless Docker: https://get.docker.com/rootless
- Usernetes: https://github.com/rootless-containers/usernetes
- Rootless KEP: https://github.com/kubernetes/enhancements/pull/1371

Questions?

- Ask me questions at #2-kubecon-maintainer (https://slack.cncf.io)


KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual