Container Plumbing Days (Apr 15, 2024)

NTT

Usernetes Gen2

Kubernetes in Rootless Docker, with Multiple Nodes

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

Akihiro Suda (NTT)
akihiro.suda.cz@hco.ntt.co.jp

Container Plumbing Days (Apr 15, 2024)

NTT

Usernetes Gen2

Kubernetes in Rootless Docker, with Multiple Nodes

Podman, and nerdctl

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

Akihiro Suda (NTT)
akihiro.suda.cz@hco.ntt.co.jp

[Introduction] Rootless Containers

- Puts container runtimes (as well as containers) in a user namespace
  - UserNS: Linux kernel's feature that maps a non-root user to a fake root (the root privilege is limited inside the namespace)

- Can mitigate potential vulnerabilities of the runtimes
  - No access to read/write other users' files
  - No access to modify the kernel (e.g., to inject invisible malware)
  - No access to modify the firmware
  - No ARP spoofing
  - No DNS spoofing

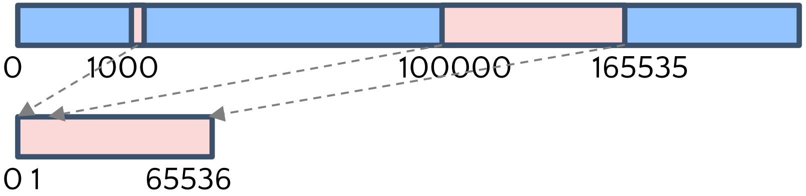
e.g., runc breakout
CVE-2024-21626
(2024-01-31)

- Also useful for shared hosts (High-performance Computing, etc.)
  - Works with GPU too

[Introduction] User namespaces

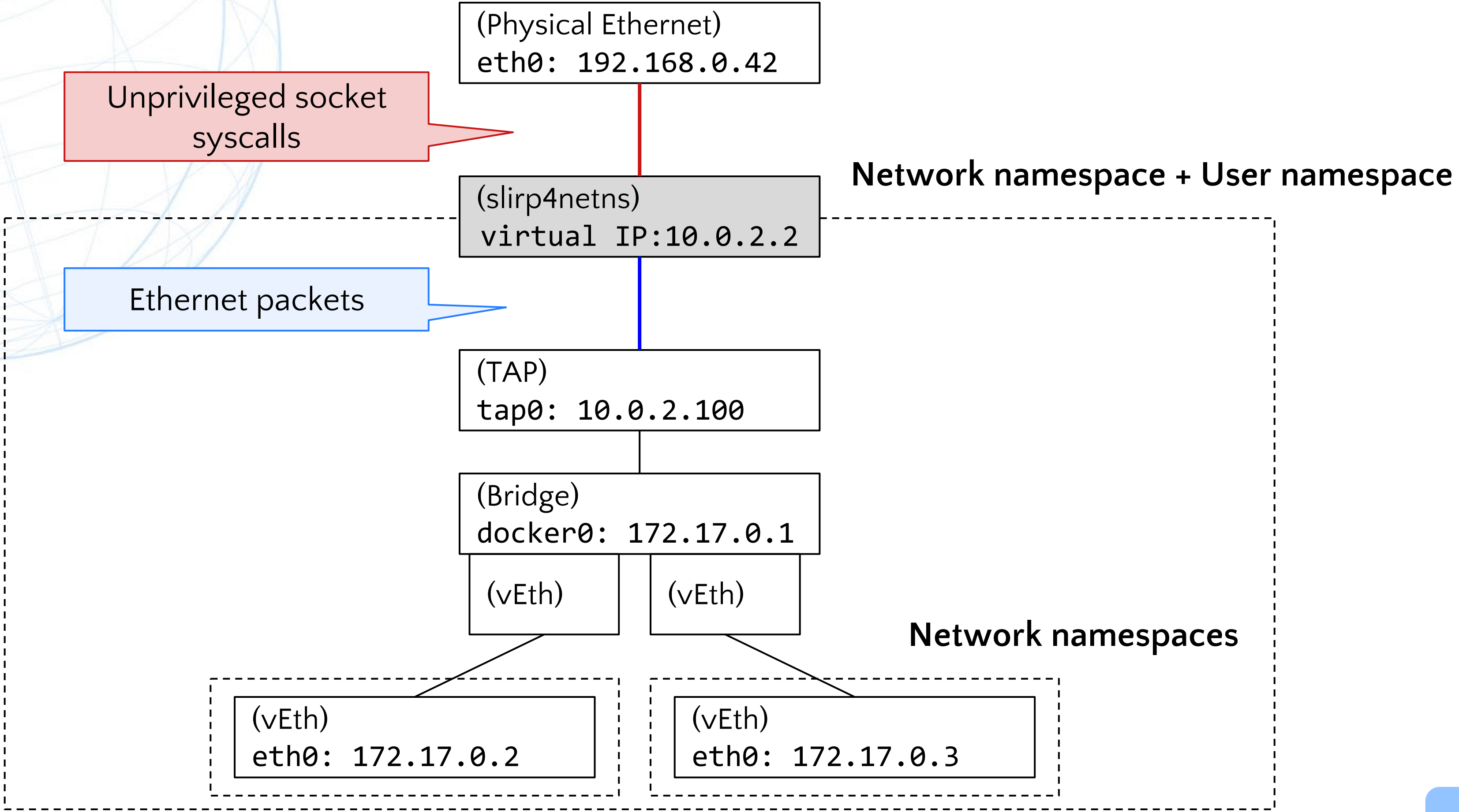
- Linux kernel’s feature to remap UIDs and GIDs

```
# /etc/subuid
1000:100000:65536
```



- UID=1000 gains **fake** root privileges (UID=0) that are enough to create containers
- The privileges are limited inside the namespace
- No privilege for setting up vEth pairs with “real” IP addresses;
user mode TCP/IP (e.g., slirp4netns) is used instead
- Also notorious as the culprit of the several kernel CVEs,
but at least it is more secure than just running everything as the root
    - Ubuntu 24.04 disables UserNS by default with the allowlist (AppArmor profiles)

[Introduction] Network namespaces



Rootless Kubernetes

- **Usernetes**: Rootless Kubernetes, since 2018
  `https://github.com/rootless-containers/usernetes`

- As old as Rootless Docker (pre-release at that time) and Rootless Podman

- The changes to upstream was merged in Kubernetes v1.22 (2021)
  - Feature gate: `KubeletInUserNamespace` (Alpha)
  - The feature gate is also used by `kind`, `minikube`, `k3s`, etc.

- The first generation ("Gen1", 2018-2023) of Usernetes didn't gain much popularity due to its complexity ("The Hard Way")

KubeletInUserNamespace feature gate

- The gate is slightly misnomer; as it requires CRI, OCI, CNI, and kube-proxy to be in the same UserNS too

- Quite "boring" gate to allow trivial permission errors
  https://github.com/search?q=repo%3Akubernetes%2Fkubernetes%20KubeletInUserNamespace&type=code
  - `dmesg`
  - `sysctl -w vm.overcommit_memory`
  - etc.

- The UserNS has to be created by an external runtime
  - **Usernetes Gen1**: RootlessKit
  - **Usernetes Gen2**: Rootless Docker/Podman/nerdctl
  - LXD/Incus can be used too

Usernetes Gen 1 vs Gen 2

”The Hard Way”

Similar to `kind` and minikube, but supports real multi-node

	Gen 1 (2018-2023)	Gen 2 (2023-)
Host dependency	RootlessKit	Rootless Docker, Rootless Podman, or Rootless nerdctl (contaiNERD CTL)
Supports kubeadm	No	Yes
Supports multi-node	Yes, but practically No, due to complexity	Yes
Supports hostPath volumes	Yes	Yes, for most paths, but needs an extra config

File layout

Everything is just a plain text file, for ease of customization

- `Makefile`
  - Defines targets like `make up` to wrap `docker compose up`, etc.
- `Dockerfile`
  - `FROM kindest/node` (`kind`'s node image) with a few additional `ADD` and `RUN`
- `docker-compose.yaml`
  - Just defines a single node container
  - Currently, node ports, etc. have to be statically defined here
- `kubeadm-config.yaml`
  - Configures feature gates, CIDRs, TLS SANs, etc.

Usage

```
# Bootstrap the first node
make up
make kubeadm-init
make install-flannel
```

```
# Enable kubectl
make kubeconfig
export KUBECONFIG=$(pwd)/kubeconfig
kubectl get pods -A
```

```
# Multi-node
make join-command
scp join-command another-host:~/usernetes
ssh another-host make -C ~/usernetes up kubeadm-join
make sync-external-ip
```

Multi-node Network

- VXLAN is known to work
  - Just `kubectl -f kube-flannel.yaml`

- “External IP” is used, as the containerized kubelet’s IP is not accessible from other nodes
  - `kubelet` is launched with `--cloud-provider=external`
  - `node.status.addresses` is dynamically patched with `kubectl patch node`
  - `node` is also annotated with `flannel.alpha.coreos.com/public-ip-overwrite`
  - UDP checksums are recomputed with `ethtool -K flannel.1 tx-checksum-ip-generic off`

Experimental: network acceleration

- Bypass4netns allows bypassing slirp4netns to eliminate the overhead caused by the usermode TCP/IP
  https://github.com/rootless-containers/bypass4netns

- Captures socket syscalls inside the NetNS, reconstructs the FDs outside the NetNS, and replaces the FDs inside the NetNS, using `seccomp_unotify(2)`

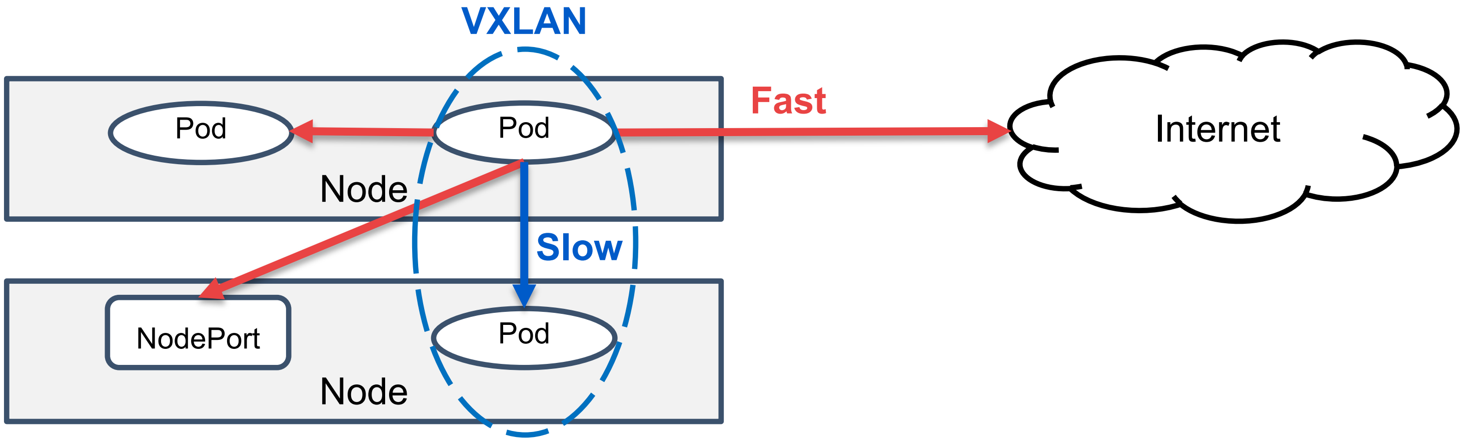
- As fast as the host network (e.g., 1.28 Gbps vs 49.9 Gbps)

- Bypass4netns supports both `connect(2)` and `bind(2)`,
  but Usernetes only supports accelerating `connect(2)` currently
  - `bind(2)` is already fast anyway

- Available for nerdctl

Experimental: network acceleration

- Pod-to-Pod communications across multiple nodes are not accelerated yet
  - VXLAN packets are generated by the kernel itself and cannot be intercepted via `seccomp_unotify(2)`
  - NodePorts can be still accelerated, as it does not incur VXLAN packets



Experimental: network acceleration

NTT

- iperf3 (TCP) benchmark across multiple nodes

	slirp4netns	bypass4netns
Pod → Pod (same node)	37.6 Gbps	37.6 Gbps
Pod → Pod (different node)	1.40 Gbps	1.41 Gbps
Pod → NodePort (same node)	1.28 Gbps	**49.9 Gbps**
Pod → NodePort (different node)	1.47 Gbps	**9.53 Gbps**
Host → NodePort (same node)	50.2 Gbps	49.4 Gbps
Host → NodePort (different node)	9.53 Gbps	9.52 Gbps

IaaS: Amazon EC2 (m7i.2xlarge)

Versions: Ubuntu 22.04, nerdctl v2.0.0-beta.4, bypass4netns v0.4.1, Usernetes Gen2-v20240410.0 (Kubernetes v1.29)

https://github.com/rootless-containers/usernetes/discussions/329

Future works

- Integrate bypass4netns into Docker and Podman too

- Support accelerating Pod-to-Pod communications across different nodes, perhaps with a sidecar proxy that would forward packets to NodePorts

- Support dynamic port forwarding
  – Ports are currently statically defined in `docker-compose.yaml`
  – If `docker container update` could support modifying port forwards, Usernetes coud just watch Kubernetes service events and update the Docker ports accordingly https://github.com/docker/cli/issues/5013

- Help other Kubernetes distributions to support rootless
  – k3s has been supporting rootless since 2019, but still lacks support for multi-node setup
  – Are Podman folks interested in running OKD inside Rootless Podman?