hackerspace.gr

Linux Kernel Namespaces

(an intro to soft-virtualization)

kargig [at] void.gr
@kargig

GPG: 79B1 9198 B8F6 803B EC37 5638 897C 0317 7011 E02C

whoami

System & services engineer @ GRNET

Messing with Linux, Security, Privacy, IPv6

Namespaces

Present different view of the system to different processes

→ Isolation

NOT a new idea → Plan9 (1992)

Types of Linux Kernel Namespaces

Currently 6+3 different types

Significant changes:

- User namespaces (2.6.23 ~ 2008)
- Unprivileged process can create a new namespace in which it has full (root) privileges (3.8)

hackerspace.gr

The 9 types

Mount

UTS

IPC

PID

Network

User

Syslog

Audit

Cgroup

Mount

- Introduced: 2.4.19
- Purpose: Different processes have different views of the mount points → “next-gen chroots”
- Short name: mnt

```
# propagation between host & namespaces
mount --make-(r)shared  / (2-way sharing)
mount --make-(r)private / (no sharing)
mount --make-(r)slave   / (1-way sharing)
```

(*) systemd uses shared flag by default

hackerspace.gr

UTS

- Introduced: 2.6.19
- Purpose: each namespace can have different hostname + domainname
- Short name: uts

(*) UTS = Unix Timesharing System

hackerspace.gr

IPC

- Introduced: 2.6.19
- Purpose: IPC Isolation
  - POSIX msg queue isolation in (2.6.30+)
- Short name: ipc

hackerspace.gr

PID

- Introduced: 2.6.24
- Purpose: processes in different namespaces can have the same pid
  - pid inside ns != pid outside ns
  - Each container (*) can have its own init (pid 1)
  - Multiple ns create multiple nested process trees
  - Migrate containers (*) across hosts keeping the same internal pids
- Short name: pid

hackerspace.gr

Network

- Introduced: 2.6.24
- Purpose: different network devices, IP addresses, routing tables,etc per namespace
  - Containers w/ network capabilities
- Short name: net

hackerspace.gr

User

- Introduced: 2.6.23 (~3.8)
- Purpose: isolate {u,g}id space. Different uid inside and outside a namespace
  - 3.8+ unprivileged users can create user namespaces (and have root inside)
- Short name: user

hackerspace.gr

Syslog

- Introduced: 3.9
- Purpose: different kernel messages per namespace

Audit

- Introduced: 3.9
- Purpose: different audit subsystem messages per namespace

hackerspace.gr

Cgroups

- Introduced: 3.9
- Purpose: different cgroup hierarchy per namespace

hackerspace.gr

Kernel API

clone()

- Create a child process, like fork() with CLONE_NEW(*) flags

setns()

- Add process to a namespace

unshare()

- Like clone() but for the calling process()

(*) CLONE_NEWIPC, CLONE_NEWNS, CLONE_NEWNET, CLONE_NEWPID, CLONE_NEWUSER, CLONE_NEWUTS

Namespace info

/proc/PID/ns/

- One file (symlink) for each namespace type
- Way to discover if two processes are in the same namespace

```
# ls -Fla /proc/2957/ns/
lrwxrwxrwx 1 root root 0 Nov  7 16:57 ipc -> ipc:[4026532507]
lrwxrwxrwx 1 root root 0 Nov  7 16:57 mnt -> mnt:[4026532505]
lrwxrwxrwx 1 root root 0 Nov  7 14:24 net -> net:[4026532441]
lrwxrwxrwx 1 root root 0 Nov  7 16:57 pid -> pid:[4026532508]
lrwxrwxrwx 1 root root 0 Nov  7 16:57 user -> user:[4026531837]
lrwxrwxrwx 1 root root 0 Nov  7 16:57 uts -> uts:[4026532506]
```

Demo!

Create a new Networking namespace and run a shell

(outside)# unshare --net /bin/sh

(inside)# ip link

Demo!

Create a new User namespace and run a shell

(outside)# unshare --user /bin/sh

(inside) $ ps auxww

If you want to run it as user and not as root in Debian:
cat /proc/sys/kernel/unprivileged_userns_clone
0
echo “1”>/proc/sys/kernel/unprivileged_userns_clone
(outside)$ unshare --user /bin/sh
(inside) $ ps auxww

Demo!

Create a new User+PID namespace and run a shell

(outside)# unshare --pid --fork --user /bin/sh

(inside) $ ps auxww

Demo!

Create a new User+PID namespace w/ different /proc and run a shell

(outside)# unshare --pid --fork --user --mount-proc /bin/sh

(inside) $ ps auxww

(*) Actually that's a User+PID+Mount namespace due to /proc private mount

Demo!

Create a Mount namespace with a private mount point

```
(outside)# unshare -m /bin/bash
(inside) # mount --make-rprivate /
(inside) # mydir=`mktemp -d --tmpdir=/tmp`
(inside) # mount -o size=1m -t tmpfs tmpfs $mydir
(inside) # mount
(outside)# mount
```

Demo!

Create a Network namespace

```
# ip netns add myns
# ip link add name veth0 type veth peer name veth1 netns myns
# ip netns exec myns ip link
# ip netns exec myns ip link set dev lo up
# ip netns exec myns ip a add 10.0.0.2/24 dev veth1
# ip netns exec myns ip link set veth1 up
# ip a add 10.0.0.1/24 dev veth0
# ip link set dev veth0 up
# ip netns exec myns ip a
# ip netns exec myns ping 10.0.0.1
```

Using namespaces

Firejail

Description: “SUID sandbox […] using Linux namespaces, seccomp-bpf and Linux capabilities”

- Create ad-hoc sandboxes, eg for browsers, mail clients or to test new daemons
- Has a GUI to view/edit sandboxes

systemd-nspawn

Systemd-nspawn (pid,mount,ipc,uts namespaces)

spawn a container (way better than chroot/schroot)

```
# systemd-nspawn -D /usr/src/sid/
```

boot into a container

```
# systemd-nspawn -b -D /usr/src/stretch/
```

Moar systemd

Haters gonna hate

Systemd.exec

ReadWriteDirectories=/var/www/

ReadOnlyDirectories=/var/www/mywebsite

InaccessibleDirectories=/srv/private

PrivateTmp=(Bool) → mount ns

PrivateDev=(Bool) → mount ns

PrivateNetwork=(Bool) → network ns

ProtectSystem=Bool or 'full' → mount ns (ro /usr /boot + /etc)

Container

For you hipster people

Docker, LXC, Kubernetes, Mesos, CoreOS, whatever_else_came_out_just_this_morningOS

Yes, they use Linux Kernel Namespaces...

RancherVM: KVM inside Docker... really...yes REALLY...

hackerspace.gr

Hard to soft isolation

Physical machines → Virtual machines → Containers → Namespaces → chroot → no isolation

A **container** is a collection of namespaces

A **namespace** is a specific resource type that can be split up and partitioned

Security...

Important CVEs

CVE 2013-1858 user escalation to root

CVE-2015-2925 escape bind mount, possibly gaining root

Multiple other vulnerabilities leading to DOS/crashes

Not all fixes in Linux kernel get CVEs, multiple fixes are about kernel namespaces components...Live with it...

Ideas

Shared hosting w/ nginx + php-fpm + different user,pid,ipc,mount ns per website

Isolate + auto-Torify network clients w/ different user,pid,ipc,mount,net ns

Resources

https://lwn.net/Articles/531114/

http://www.landley.net/kdocs/ols/2006/ols2006v1-pages-101-112.pdf

http://www.haifux.org/lectures/299/netLec7.pdf

http://doger.io/

http://0pointer.de/blog/projects/changing-roots.html

Outro

Thank you!

Questions ?

kargig [at] void.gr

@kargig

GPG: 79B1 9198 B8F6 803B EC37 5638 897C 0317 7011 E02C