Sandbox-Verfahren für GNU/Linux

Programme in abgeschotteten Umgebungen ausführen

Agenda

- **Theorie / Nutzen**
- **Einzelne Verfahren**
  - Chroot
  - systemd-nspawn
  - Linux Container (LXC)
  - Bubblewrap
  - Firejail
- **Demonstation von Firejail**

Agenda

- **Theorie / Nutzen**
- **Einzelne Verfahren**
  - Chroot
  - systemd-nspawn
  - Linux Container (LXC)
  - Bubblewrap
  - Firejail
- **Demonstration von Firejail**

Theorie / Nutzen

- **Üblich:** Anwendung läuft mit Rechten des Nutzers
  - Hat damit **uneingeschränkten Zugriff** auf alle Dateien und Verzeichnisse des Nutzers
  - Sieht alle Prozesse des Nutzers
  - Hat uneingeschränkt Zugriff aufs Netzwerk
  - Kann alle Userspace-Funktionen des Kernels nutzen, obwohl oft nur ein Teil davon nötig

Theorie / Nutzen

- **Wunsch:** Programme in einer eingeschränkten Umgebung ausführen
  - Nur noch Zugriff auf benötigte Dateien und Verzeichnisse
  - Netzwerkzugriff einschränken
  - Sicht auf andere Prozesse einschränken
  - Nutzung von Kernel-Funktionen (Systemaufrufe) auf das Notwendige beschränken

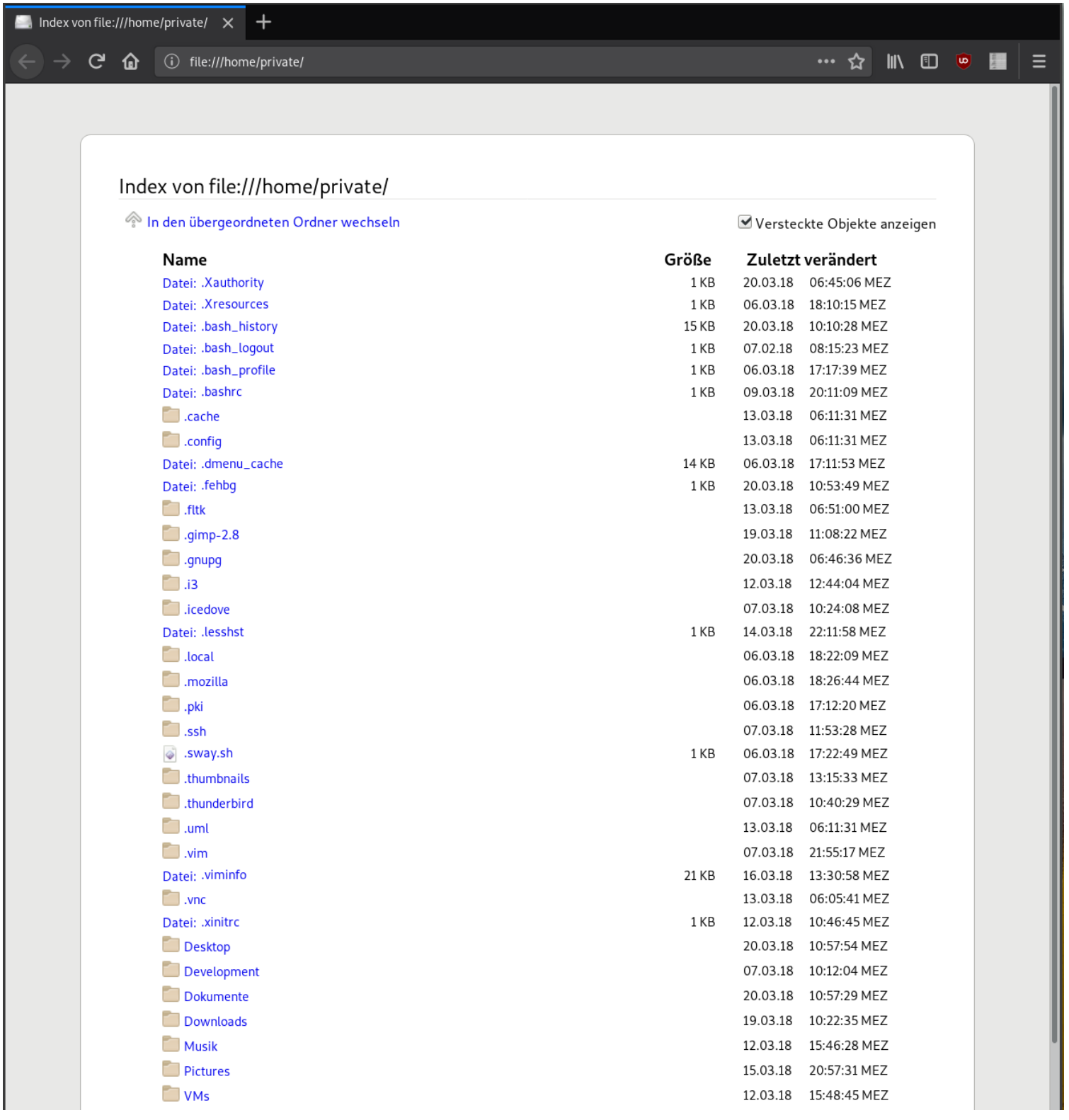
Beispiel Steam

- Steam ist eine Internet-Vertriebsplattform für Computerspiele
- Kauf, Update und Überwachung (DRM) von Spielen möglich
- Seit 2013 auch für Linux erhältlich
- **Proprietär, lädt Code nach und kann diesen ausführen**
- … und hat Zugriff auf *~/.gnupg*, *~/.keepas*, *~/.bash_history*

Beispiel Firefox

- Browser bieten große Angriffsfläche durch hunderte Features und riesige Codebasis
- Führen nicht vertrauenswürdige JavaScript Programme aus
- 2016 gab es **133 Sicherheitslücken** in Firefox, davon konnten **53 zur Ausführung von Code** genutzt werden

  Quelle: https://www.cvedetails.com/product/3264/Mozilla-Firefox.html?vendor_id=452
- … und hat Zugriff auf *~/.gnupg*, *~/.keepas*, *~/.bash_history*


Index von file:///home/private/
file:///home/private/
Index von file:///home/private/
In den übergeordneten Ordner wechseln
Versteckte Objekte anzeigen
Name Größe Zuletzt verändert
Datei: .Xauthority 1 KB 20.03.18 06:45:06 MEZ
Datei: .Xresources 1 KB 06.03.18 18:10:15 MEZ
Datei: .bash_history 15 KB 20.03.18 10:10:28 MEZ
Datei: .bash_logout 1 KB 07.02.18 08:15:23 MEZ
Datei: .bash_profile 1 KB 06.03.18 17:17:39 MEZ
Datei: .bashrc 1 KB 09.03.18 20:11:09 MEZ
.cache 13.03.18 06:11:31 MEZ
.config 13.03.18 06:11:31 MEZ
Datei: .dmenu_cache 14 KB 06.03.18 17:11:53 MEZ
Datei: .fehbg 1 KB 20.03.18 10:53:49 MEZ
.fltk 13.03.18 06:51:00 MEZ
.gimp-2.8 19.03.18 11:08:22 MEZ
.gnupg 20.03.18 06:46:36 MEZ
.i3 12.03.18 12:44:04 MEZ
.icedove 07.03.18 10:24:08 MEZ
Datei: .lesshst 1 KB 14.03.18 22:11:58 MEZ
.local 06.03.18 18:22:09 MEZ
.mozilla 06.03.18 18:26:44 MEZ
.pki 06.03.18 17:12:20 MEZ
.ssh 07.03.18 11:53:28 MEZ
.sway.sh 1 KB 06.03.18 17:22:49 MEZ
.thumbnails 07.03.18 13:15:33 MEZ
.thunderbird 07.03.18 10:40:29 MEZ
.uml 13.03.18 06:11:31 MEZ
.vim 07.03.18 21:55:17 MEZ
Datei: .viminfo 21 KB 16.03.18 13:30:58 MEZ
.vnc 13.03.18 06:05:41 MEZ
Datei: .xinitrc 1 KB 12.03.18 10:46:45 MEZ
Desktop 20.03.18 10:57:54 MEZ
Development 07.03.18 10:12:04 MEZ
Dokumente 20.03.18 10:57:29 MEZ
Downloads 19.03.18 10:22:35 MEZ
Musik 12.03.18 15:46:28 MEZ
Pictures 15.03.18 20:57:31 MEZ
VMs 12.03.18 15:48:45 MEZ

Beispiel Firefox

- Braucht Firefox wirklich Zugriff auf:
  - ~/.ssh/
  - ~/.gnupg/
  - ~/.bash_history
  - …
- **Nein!** Eigentlich doch nur auf:
  - ~/.mozilla/
  - ~/Downloads/

Lösungen

- Es gibt verschiedene Technologien
- Unterschiedliche Ansätze haben unterschiedlich hohes Schutzniveau und sind verschieden komplex
- Manche Ansätze sind nicht ohne weiteres Einsetzbar (z.B. SELinux / AppArmor)
  - OS Unterstützung erforderlich
  - Nicht in jeder Distribution gut umgesetzt
- Wir beschäftigen uns im Folgenden mit **universell einsetzbaren** Lösungen

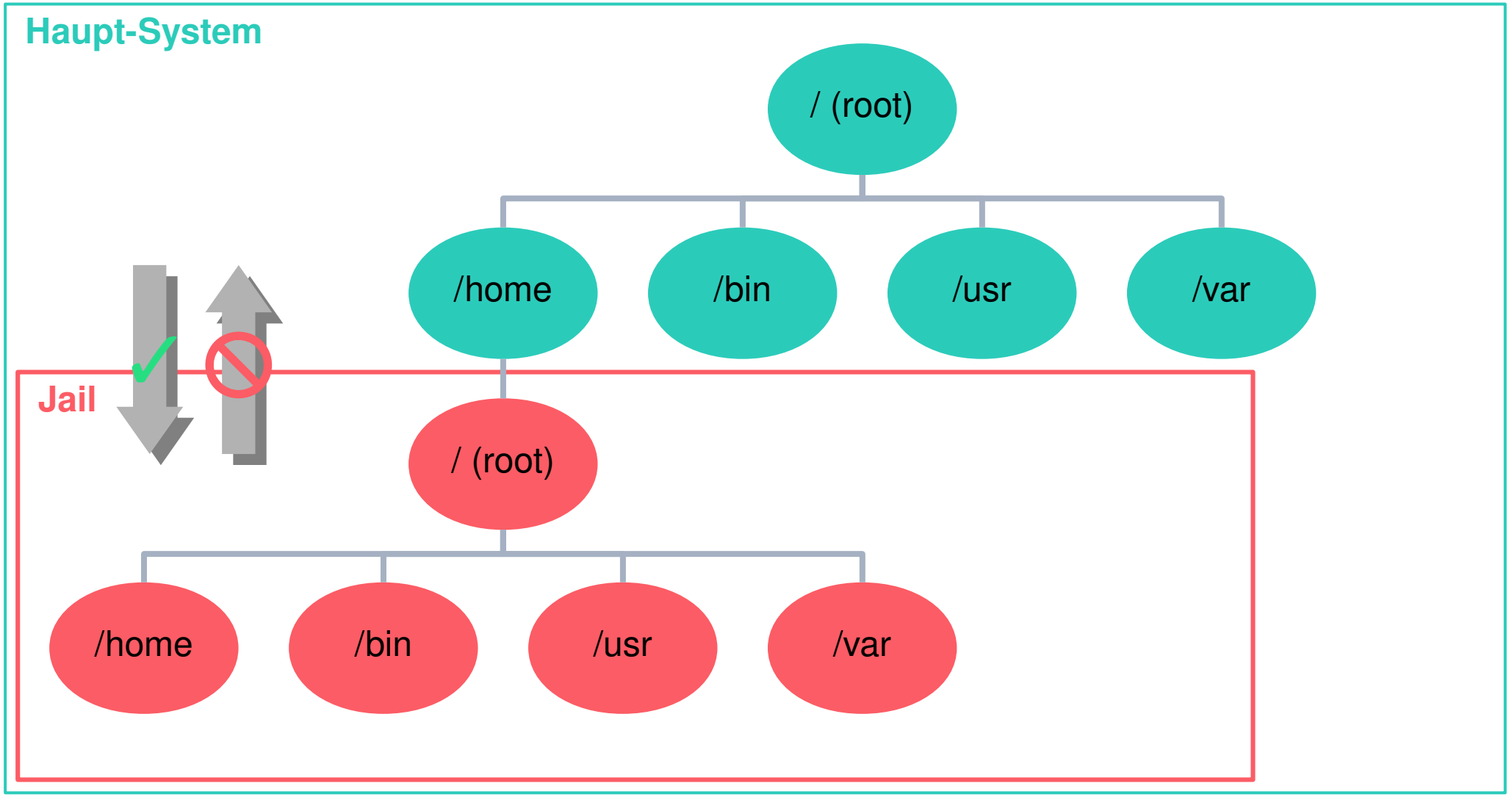
Agenda

- Theorie / Nutzen
- **Einzelne Verfahren**
  - Chroot
  - Bubblewrap
  - Linux Container (LXC)
  - systemd-nspawn
  - Firejail
- **Demonstration von Firejail**

Chroot Jail

- Sehr einfach, gibt es schon seit den 80er Jahren, sogar in UNIX-Systemen aus der „Steinzeit“
- Ursprünglich entwickelt für Testzwecke
- Verlegt Wurzeldateisystem (/) in anderes Verzeichnis. Modifikation oder Einsicht von Dateien und Verzeichnissen außerhalb ist nicht möglich.

Chroot Jail: Funktionsweise



Chroot Jail: Nachteile

- **Nachteile:**
  - **Isoliert nur Dateisystem**. Prozesse und Netzwerk-Schnittstellen weiterhin sichtbar
  - Austausch von Dateien (z.B. in ~/Downloads) zwischen Jails schwierig, **schlechte Usability**
  - Privilegierte Nutzer können unter Umständen **aus dem Jail ausbrechen**

Agenda

- Theorie / Nutzen
- **Einzelne Verfahren**
  - Chroot
  - systemd-nspawn
  - Linux Container (LXC)
  - Bubblewrap
  - Firejail
- **Demonstration von Firejail**

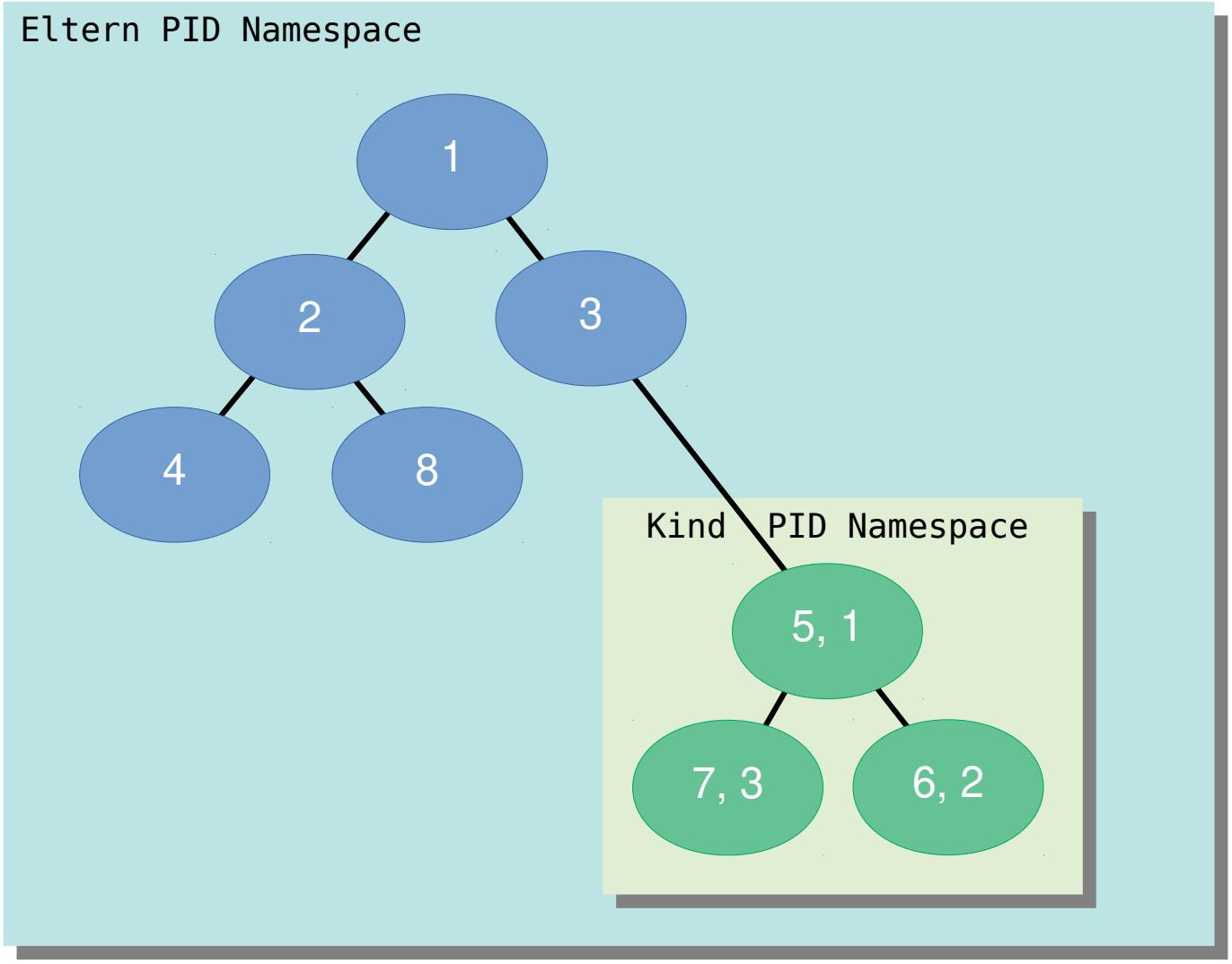
Einschub: Linux Namespaces

- Alle weiteren Verfahren basieren im Wesentlichen auf **Linux Namesapces**
- Ein Namespace (dt. Namensraum) zeigt dem laufenden Programm nur einen Teilausschnitt des Systems.
  - Individuelle Sicht auf laufende Prozesse
  - Individuelle Sicht auf Dateisystem
  - Individuelle Sicht auf das Netzwerk
  - Individuelle Sicht auf die Benutzer
  - Und weitere Sichten

Einschub: Linux Namespaces

- Dadurch ist eine umfassende Isolierung von Anwendungen möglich
- Docker/LXC und Container-Virtualisierung basieren ebenfalls auf Linux Namespaces und nutzen die gleiche Technologie
- Schrittweise seit ~2002 in Entwicklung/Einführung
  - Ursprünglich mit Mount Namespaces begonnen
  - Im Laufe der Jahre wurden weitere Namespaces hinzugefügt

Einschub: Linux Namespaces



Beispiel von Prozess ID (PID) Namespaces

- Eltern Namespace sieht Prozesse im Kindnamespace
- Im Kindnamespace nur eigne PIDs sichtbar
- PIDs im Kind-namespace unterscheiden sich vom Eltern-namespace

Einschub: Linux Namespaces

Eltern PID Namespace:

```
[jannis@ganymede ~]$ ps aux | grep bash
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
jannis   29537  0.0  0.0  14032  2248 pts/3    S    19:38   0:00 firejail --private bash
jannis   29538  0.0  0.0  14164  1728 pts/3    S    19:38   0:00 firejail --private bash
jannis   29548  0.0  0.0  25948  3984 pts/3    S+   19:38   0:00 bash
```

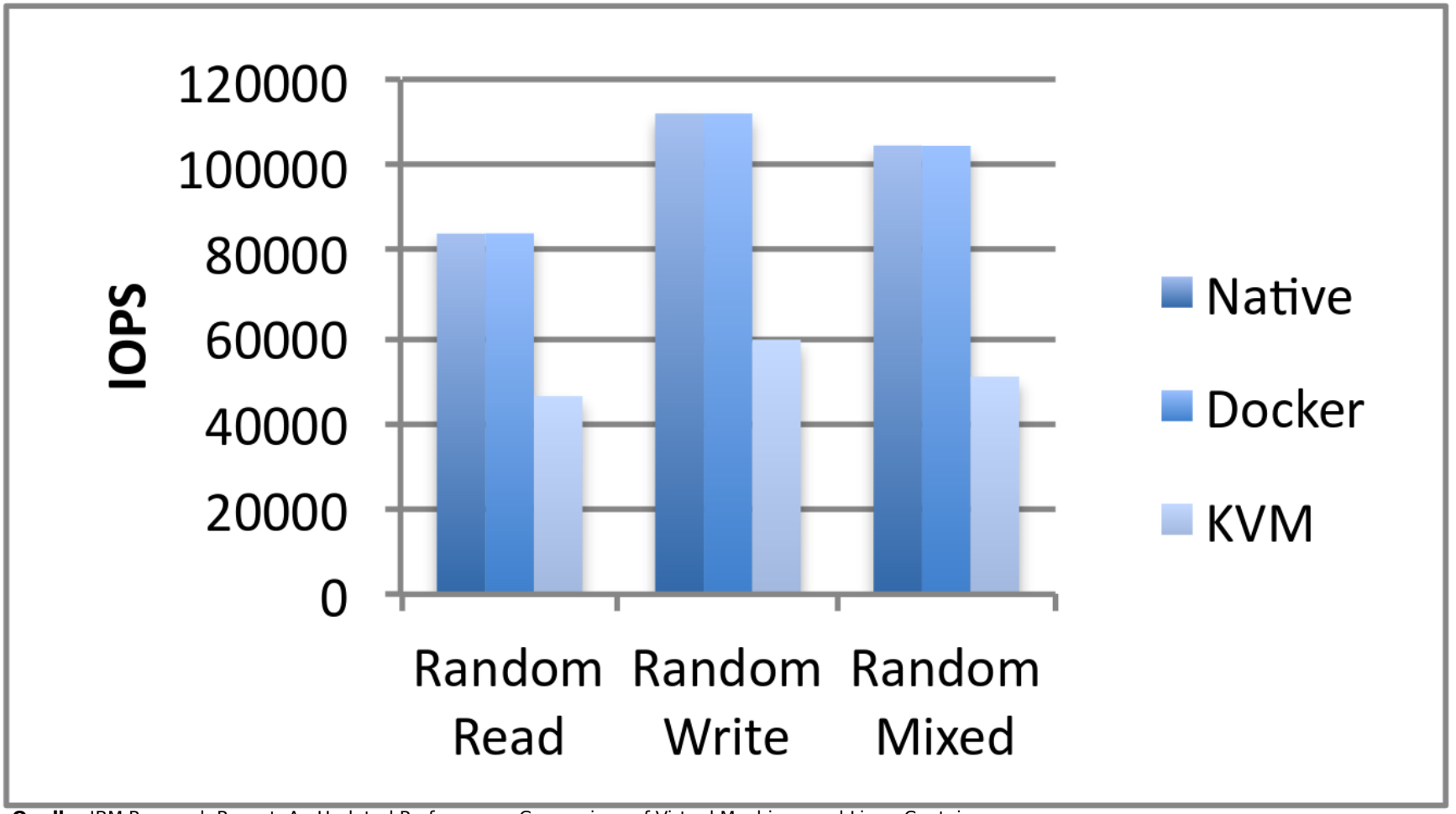
Kind PID Namespace:

```
[jannis@ganymede ~]$ firejail --private bash
[jannis@ganymede ~]$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
jannis       1  0.5  0.0  14164  1728 pts/3    S    19:38   0:00 firejail --private bash
jannis      10  0.0  0.0  25948  3848 pts/3    S    19:38   0:00 bash
jannis      11  0.0  0.0  42012  3444 pts/3    R+   19:38   0:00 ps aux
```

Einschub: Linux Namespaces

- Linux Namespaces haben, anders als virtuelle Maschinen, **keinen Einfluss auf die Systemleistung**
- Linux System startet mit einem Namespace von jedem Typ, in denen alle Prozesse laufen
  - Prozesse können zusätzliche Namespaces erzeugen
  - Oder Namespaces zusammenführen

Einschub: Linux Namespaces



Quelle: IBM Research Report, An Updated Performance Comparison of Virtual Machines and Linux Containers

systemd-nspawn

- „chroot on steroids“ – *Lennart Poettering*
- Entwickelt um systemd zu kompilieren, testen und zu debuggen
- Isoliert nicht nur Dateisystem (wie chroot), sondern **nutzt Linux Namespaces** um Sicht auf Prozesse und Netzwerk einzuschränken
- Read-only Zugriff auf Kernel-Schnittstellen wie `/sys`, `/proc/sys` und `/sys/fs/selinux`

systemd-nspawn

- Eignet sich **für Software-Entwickler** zum Testen in verschiedenen Umgebungen als leichte **Alternative zu Docker**
- Nicht zum Deployen von Software in Produktiv-Umgebung gedacht! (Dafür eher Docker/LXC)

systemd-nspawn

- Startet eigene systemd Instanz
- Dateisystem, ähnlich wie bei chroot, in einem vollständig isolierten Verzeichnis
- Ausführen von Desktop-Anwendungen möglich, aber nicht explizit vorgesehen
- Nutzungsbeispiel unter Debian

```
# debootstrap unstable ~/debian-tree/
# systemd-nspawn -D ~/debian-tree/
```

systemd-nspawn

- **Highlight:** Flüchtiger Snapshot des Systems erzeugen und damit arbeiten

```
# systemd-nspawn -D / -xb
```

- Snapshot wird samt Änderungen sofort gelöscht, wenn Container verlassen wird
- **Aber:** Nur sinnvoll, wenn / (root) auf einer btrfs Partition liegt (Snapshot Funktion)
- **Sonst:**

```
Failed to create snapshot /.#machine.03c3c0d78af586f2
from /: No space left on device
```

Agenda

- Theorie / Nutzen
- **Einzelne Verfahren**
  - Chroot
  - systemd-nspawn
  - Linux Container (LXC)
  - Bubblewrap
  - Firejail
- **Demonstration von Firejail**

Linux Container (LXC)

- Weiteres Verfahren zur Virtualisierung
- Linux Container sind **keine virtuelle Maschinen**
  - Container **nutzen den Host-Kernel**
  - Daher gibt es nur *Linux Container* unter Linux
  - Eignen sich für **Server-Anwendungen**
  - Benutzt ebenfalls Linux Namespaces zur Isolierung und Cgroups (Ressourcenverwaltung)
- Docker benutzt u.a. LXC, aber auch systemd-nspawn und weitere Techniken

Agenda

- Theorie / Nutzen
- **Einzelne Verfahren**
  - Chroot
  - systemd-nspawn
  - Linux Container (LXC)
  - Bubblewrap
  - Firejail
- Demonstration von Firejail

Bubblewrap

- „Bubblewrap“ zu deutsch „Luftpolsterfolie“
- Wird u.a. bei Flatpak verwendet
  - Flatpak „Apps für Linux“
  - Bringt Bibliotheken und Abhängigkeiten mit
  - Isoliert Anwendung von System mit Bubblewrap
- Weniger Funktionen als Firejail, denn Fokus liegt stark auf **Sicherheit**

Bubblewrap

- `$ bwrap --ro-bind /usr /usr --symlink usr/lib64 /lib64 --proc /proc --dev /dev --unshare-pid bash`
- Startet bash in einer isolierten Umgebung

```
bash-4.4$ ls -la
insgesamt 4
drwxr-xr-x   5 1000 1000  120  3. Apr 17:01 .
drwxr-xr-x   5 1000 1000  120  3. Apr 17:01 ..
drwxr-xr-x   4 1000 1000  340  3. Apr 17:01 dev
lrwxrwxrwx   1 1000 1000    9  3. Apr 17:01 lib64 -> usr/lib64
dr-xr-xr-x 194    0    0    0  3. Apr 17:01 proc
drwxr-xr-x   9    0    0 4096  3. Apr 17:01 usr
bash-4.4$
bash-4.4$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
1000         1  0.0  0.0   6384   100 ?        S    17:01   0:00 bwrap --ro-bind /usr /usr --s
1000         3  0.0  0.0  24056  4176 ?        S    17:01   0:00 bash
1000         9  0.0  0.0  37536  3248 ?        R+   17:02   0:00 ps aux
```

Agenda

- Theorie / Nutzen
- **Einzelne Verfahren**
  - Chroot
  - systemd-nspawn
  - Linux Container (LXC)
  - Bubblewrap
  - Firejail
- Demonstration von Firejail

Firejail

- Bietet umfassende Sandbox Möglichkeiten
- Nutzt Linux Namespaces, Seccomp Filter und Linux Capabilities
- **Sehr einfach** in der Handhabung
- Bringt **über 400 vorgefertigte Profile** für populäre Anwendungen mit
- Zum Isolieren von Desktop-Anwendungen gedacht (kann mit X11 und PulseAudio umgehen)

Firejail

- Profile liegen in `/etc/firejail/`
- Da als SUID-Programm implementiert, lassen sich Container auch ohne root/sudo erzeugen
- Starten einer **flüchtigen** und **leeren** Firefox-Instanz:
  ```
  $ firejail --private firefox
  ```
- Testen eines Profils mit bash:
  ```
  $ firejail --profile=/etc/firejail/firefox.profile bash
  ```

Firejail: Profile

- Firejail können alle Parameter/Einstellungen per Kommandozeile übergeben werden
- Bei komplexen Umgebungen ist das aber unhandlich, da langer Befehl
- Daher gibt es Profile, in denen alle Einstellungen für ein jeweiliges Programm vorgenommen werden können

Firejail: Profile

- In Profilen wird festgelegt
  - Auf welche Dateien/Verzeichnisse wie zugegriffen werden kann
  - Welche zusätzlichen Sicherheitsfunktionen aktiviert werden
  - Welche Seccomp-Filter aktiviert werden
  - Ob eine Shell zur Verfügung steht
  - uvm.

Firejail: Profile

```
# Firejail profile for vlc
# This file is overwritten after every install/update
# Persistent local customizations
include /etc/firejail/vlc.local
# Persistent global definitions
include /etc/firejail/globals.local
noblacklist ${HOME}/.config/vlc
noblacklist ${HOME}/.local/share/vlc

include /etc/firejail/disable-common.inc
include /etc/firejail/disable-devel.inc
include /etc/firejail/disable-passwdmgr.inc
include /etc/firejail/disable-programs.inc

include /etc/firejail/whitelist-var-common.inc

caps.drop all
netfilter
# nogroups
nonewprivs
noroot
protocol unix,inet,inet6,netlink
seccomp
shell none

private-bin vlc,cvlc,nvlc,rvlc,qvlc,svlc
private-dev
private-tmp

# mdwe is disabled due to breaking hardware accelerated decoding
# memory-deny-write-execute

noexec ${HOME}
noexec /tmp
```

Agenda

- Theorie / Nutzen
- **Einzelne Verfahren**
  - Chroot
  - systemd-nspawn
  - Linux Container (LXC)
  - Bubblewrap
  - Firejail
- **Demonstration von Firejail**

Demonstation von Firejail