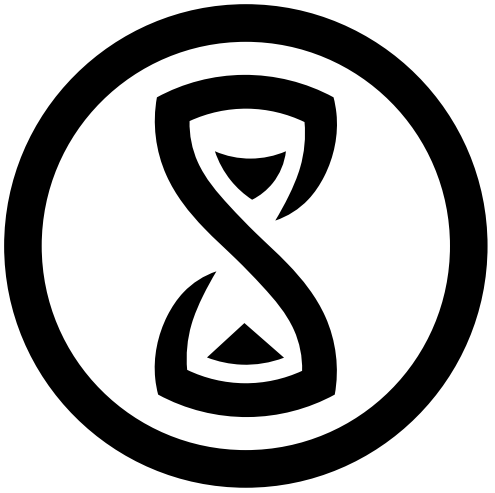
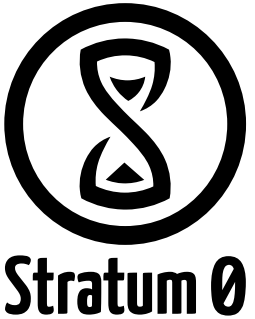
Sandboxing von systemd-Diensten

Aka: Poor man's docker

Joke
14.04.2023

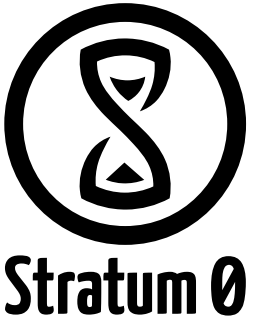


Motivation



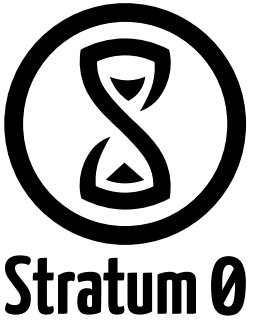
- Ich war neulich auf einer Weiterbildung
- Man kann mit systemd Diensten sehr gut Rechte wegnehmen
- Relativ wenig Aufwand für low-hanging fruits
- Funktioniert auch ohne AppArmor/SELinux und Co
- `Wird erschreckend wenig gemacht`
- `Das will ich ändern`

Was braucht es?



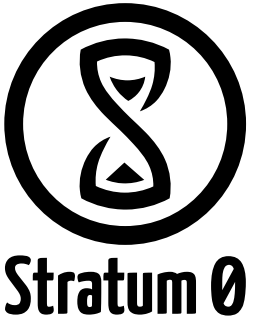
- Basiert auf gleichen Kram wie Docker und co (namespaces, control groups etc)
- Hinreichend aktuelles systemd (entsprechender Code in systemd-analyze tauchte erstmals 2018 auf, wurde seitdem immer wieder erweitert)
- Debian Bullseye kann das, also auch Ubuntu 20.04/22.04
- Debian Buster wohl nicht (Ubuntu 18.04 jedenfalls nicht und das basiert auf Buster)
- In der Redhat Welt: Ab RockyLinux8 geht es auf jeden Fall
- Rest: Kein Plan

Vorgehensweise



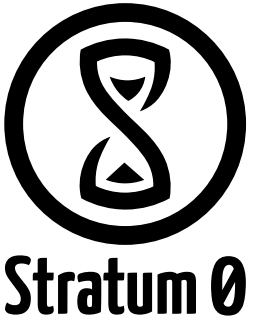
- systemd-analyze security gibt Übersicht, wie weit Dienste das nutzen
- Achtung: Das ist kein generisches Security-/Schwachstellenscanner etc!
- Es überprüft lediglich, wie weit die Sandboxing-Funktionen genutzt werden oder eben nicht
- Relevante manual pages:
  - systemd.analyze(1)
  - systemctl(1)
  - systemd.exec(5)
  - systemd.resource-control(5)
  - capabilites(7)

Legen wir los



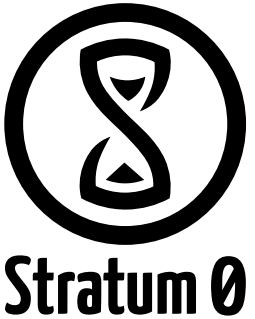
- Neuen Dienst anlegen:
  systemctl edit --force --full dienstname.service
- Bestehenden Dienst bearbeiten:
  - systemctl edit --full dienstname.service
  - Lässt man –-full weg, bearbeitet man stattdessen overrides
- Dienst anzeigen und testen:
  - systemctl cat dienstname.service
  - systemd-analyze security dienstname.service
  - systemctl start dienstname.service

Capabilities einschränken



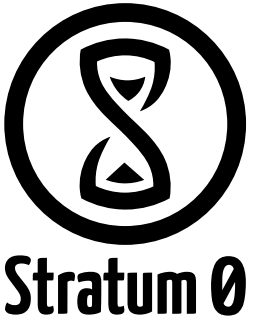
- capabilities(7) sind seit 2.2 im Linux Kernel
- Hintergrund: Klassische Trennung zwischen priviligierten (root) und unpriviligierten (alle anderen) Prozessen nicht so dolle
- Capabilities erlauben weitere Granualität
- Im system vorhandene anzeigen:
  - systemd-analyze capabilities
- Systemd erlaubt es den Diensten explizit welche wegzunehmen und zu genehmigen:
  - AmbientCapabilities Zusätzliche erlauben
  - CapabilityBoundingSet: Capabilites wegnehmen

SystemCalls einschränken



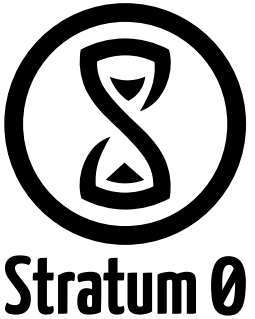
- Man kann den Diensten explizit das Ausführen bestimmter Systemcalls erlauben oder verbieten
- Problem: Woher weiß ich, welche ich nehmen soll?
- Gibt glücklicherweise ein paar sets mit (mehr oder weniger) sinnvollen Defaults:
  - @clock für Änderungen an der Systemzeit
  - @file-system analog für alles mit Dateizugriffen
  - @obsolete Obsolete, seltene oder gar nicht implementierte Syscalls
  - @resources Syscalls für Änderungen am Scheduling und Ressourcenlimits
  - @privileged Alle die Superuser-Rechte benötigen
  - @system-service Wichtigster Kram für Dienste
- Im Idealfall fängt man damit an, dass man @system-service erlaubt und @privileged sowie @resources verbietet
- Wenn man glück hat, geht noch alles, wenn nicht muss man halt nach und nach wieder Sachen erlauben ;)
- Als Entwickler/in sollte man davon eh eine genaue Vorstellung haben
- Liste ist nicht vollständig, genaueres:
  - sudo systemd-analyze syscall-filter
  - systemd.exec(5)

Weitere Nettigkeiten (die ich noch nicht ausprobiert habe...)



- DynamicUser: Erzeugt zur Laufzeit User/Gruppe, die nur im RAM und nur für den Dienst existieren
- PrivateNetwork=true Dienst hat eigenen Network namespace mit nur einem loopback Interface
- RootDirectory: chroot
- RootImage: mountet block oder loopback device und chrootet dann darein
- MountImages: Analog zu RootImage nur ohne loopback
- BindPaths=, BindReadOnlyPaths= bind mounts
- SELinuxContext=,AppArmorProfil= AppArmor-Profil oder SELinuxContext für Dienst setzen
- systemd-container mit systemd.nspawn
- Diverse andere Sachen ;)

Zum Weiterlesen



- Manual pages:
  - systemd.analyze(1)
  - systemctl(1)
  - systemd.exec(5)
  - systemd.resource-control(5)
  - capabilites(7)
- Internet (Blogposts und co):
  - Systemd Hardening allgemein:
    - https://www.flashsystems.de/articles/systemd-hardening/
    - https://www.linuxjournal.com/content/systemd-service-strengthening
  - Erklärung der Optionen für Capabilities:
    - https://unix.stackexchange.com/q/580597
- Vorschlag für Standard-Pattern plus Erklärungen: https://gist.github.com/ageis/f5595e59b1cddb1513d1b425a323db04
  - Gibt noch mehr Erklärungen plus Beispiele, sowohl bei github als auch sonstwo →Google systemd hardening
- Beispiel für container mit systemd-nspawnd: Ein Gopher-Server in LUA https://strotmann.de/archive.html → Dort die Postings zu gophermoon

Danke für eure Aufmerksamkeit!

Slides und Beispiele:
https://github.com/johannesst/systemd-hardening

Stratum 0 e.V. Braunschweig
https://stratum0.org