LINUX SERVER SICHERHEIT

KURSNOTIZEN

SEPTEMBER/NOVEMBER 2025

CARSTEN STROTMANN

Version 20251128, 28.11.2025

INHALTSVERZEICHNIS

CHAPTER 1. INFO

- Live Script: https://server.linux-sicherheit.org [https://server.linux-sicherheit.org]
- Einführung und Krypto Basics: https://server.linux-sicherheit.org/crypto.html [https://server.linux-sicherheit.org/crypto.html]
- Linuxhotel Lab-Server (Bewertung, Passwort ändern etc): https://lab.linuxhotel.de [https://lab.linuxhotel.de]
- Wiki https://wiki.linuxhotel.de/doku.php?id=server-sicherheit:start [https://wiki.linuxhotel.de/doku.php?id=server-sicherheit:start]
- Literaturliste
  - SUDO Mastery 2nd Edt. https://mwl.link/sudo-mastery-2nd-edition.html [https://mwl.link/sudo-mastery-2nd-edition.html]
  - SSH Mastery 2nd Edt. https://mwl.link/ssh-mastery-2nd-edition.html [https://mwl.link/ssh-mastery-2nd-edition.html]
  - PAM Mastery https://mwl.link/pam-mastery.html [https://mwl.link/pam-mastery.html]
  - PGP & GPG https://www.michaelwlucas.com/nonfiction/pgp-gpg-email-for-the-practical-paranoid [https://www.michaelwlucas.com/nonfiction/pgp-gpg-email-for-the-practical-paranoid]
  - Christof Paar, Jan Pelzl□—□Understanding Cryptography ISBN: 3642446493
  - Klaus Schmeh□—□Kryptografie ISBN: 3864900158
  - Niels Ferguson, Bruce Schneier, Tadayoshi Kohno□—□Cryptography Engineering ISBN: 0470474246
  - Malcolm Harkins□—□Managing Risk and Information Security ISBN: 1430251131
  - Tobias Klein□—□Aus dem Tagebuch eines Bughunters ISBN: 978-3-89864-659-8
  - Ross Anderson□—□Security Engineering (PDF: http://www.cl.cam.ac.uk/~rja14/book.html)
  - Steven M. Bellovin□—□Thinking Security: Stopping Next Year's Hackers (Addison-Wesley Professional Computing) ISBN-10: 0134277546/ISBN-13: 978-0134277547
  - Donald A. Tevault□—□Mastering Linux Security and Hardening ISBN: 1788620305
  - Kyle Rankin□—□Linux Hardening in Hostile Networks ISBN-13: 978-0134173269
  - Nikolai Philipp Tschacher□—□Typosquatting in Programming Language Package Managers http://incolumitas.com/data/thesis.pdf [http://incolumitas.com/data/thesis.pdf]

CHAPTER 2. TAG 1/2 (SEPTEMBER)

2.1. CRYPTO WETTBEWERBE

- Cryptographic competitions http://competitions.cr.yp.to/ [http://competitions.cr.yp.to/]
- NIST SHA3 Competition http://csrc.nist.gov/groups/ST/hash/sha-3/ [http://csrc.nist.gov/groups/ST/hash/sha-3/]
- NIST AES Competition http://csrc.nist.gov/archive/aes/index2.html [http://csrc.nist.gov/archive/aes/index2.html]
- Post-Quantum Cryptography Standardization https://csrc.nist.gov/pqc-standardization [https://csrc.nist.gov/pqc-standardization]

2.2. LINKS ZUM THEMA KRYPTO-ALGORITHMEN

- Guidance for Choosing an Elliptic Curve Signature Algorithm in 2022 https://soatok.blog/2022/05/19/guidance-for-choosing-an-elliptic-curve-signature-algorithm-in-2022/ [https://soatok.blog/2022/05/19/guidance-for-choosing-an-elliptic-curve-signature-algorithm-in-2022/]

> TL;DR
>
> Ed25519 is great. NIST P-256 and P-384 are okay (with caveats). Anything else is questionable, and their parameter selection should come with a clear justification.

2.3. AUFGABE: HASH UND HMAC

- Lade die Datei https://server.linux-sicherheit.org/LinuxSecurityEinfuehrung.pdf [https://server.linux-sicherheit.org/LinuxSecurityEinfuehrung.pdf]
- Welcher SHA256 Hash-Wert ist korrekt?
  a. 0245bcddf700ddfb4fd37c58bbfc67e19b9aa8827d0242fc9beb40078925f191
  b. f8eee6980c49d8020b10449a0fec6544e266af09
  c. 22833704a424725d5725d977cc733244afe0e38a6e2a987a560c95aa39da2d01
  d. AEGHHA91288ABCCVA005612991ZZ612566189811

```
wget https://server.linux-sicherheit.org/LinuxSecurityEinfuehrung.pdf
openssl dgst -sha256 < LinuxSecurityEinfuehrung.pdf
```

- Der folgende Wert sind die HMACs für die PDF-Datei. Welcher PSK (pre-shared-secret) wurde verwendet:

```
HMAC-SHA256 =
 3989780c03e83f553dca485c38a4d6239de58affbe27f0d7b572be00a5e512ee
```

- Auswahl der PSKs:
  a. 'LinuxHotel'
  b. 'Essen-Horst'

c. 'vogelsang'

```
openssl sha256 -hmac "<psk>" < LinuxSecurityEinfuehrung.pdf
```

2.4. VERSCHLÜSSELUNGSSYSTEME

2.4.1. OpenPGP

- OpenPGP ist ein IETF Internet Standard für Dateiverschlüsselung. Der aktuelle Standard ist RFC 9580 https://datatracker.ietf.org/doc/html/rfc9580 [https://datatracker.ietf.org/doc/html/rfc9580]

2.4.2. GnuPG/GPG – GNU Privacy Guard

- GnuPG ist eine vollständige und kostenlose Implementierung des OpenPGP–Standards gemäß RFC 4880 (auch bekannt als PGP). Mit GnuPG können Sie Ihre Daten und Ihre Kommunikation verschlüsseln und signieren. Es verfügt über ein vielseitiges Schlüsselverwaltungssystem sowie Zugriffsmodule für alle Arten von Verzeichnissen öffentlicher Schlüssel. GnuPG, auch bekannt als GPG, ist ein Befehlszeilentool mit Funktionen zur einfachen Integration in andere Anwendungen. Es steht eine Vielzahl von Frontend–Anwendungen und Bibliotheken zur Verfügung. GnuPG bietet außerdem Unterstützung für S/MIME und Secure Shell (ssh). In den letzten Jahren hat sich GnuPGP vom IETF OpenPGP Standard entfernt, GnuPGP unterstützt nur den älteren RFC 4880, nicht jedoch den neueren Standard RFC 9580. Webseite: https://gnupg.org [https://gnupg.org]

2.4.3. Sequoia–PGP

- Sequoia–PGP ist eine Implementation des OpenPGP–Standards in der Programmiersprache RUST

  Webseite: https://sequoia–pgp.org/ [https://sequoia–pgp.org/]

2.4.4. AGE – Actually Good Encryption

- Ein einfaches, modernes und sicheres Verschlüsselungstool (und eine Go–Bibliothek) mit kleinen expliziten Schlüsseln, ohne Konfigurationsoptionen und mit UNIX–ähnlicher Kombinierbarkeit: https://age–encryption.org [https://age–encryption.org]

2.4.5. XCA – X Certificate and Key Management

- XCA dient zum Erstellen und Verwalten von X.509–Zertifikaten, Zertifikatsanforderungen, privaten RSA–, DSA– und EC–Schlüsseln, Smartcards und CRLs. Alle Zertifizierungsstellen können rekursiv Unterzertifizierungsstellen signieren. Diese Zertifikatsketten werden übersichtlich dargestellt. Für eine einfache unternehmensweite Nutzung gibt es anpassbare Vorlagen, die für die Erstellung von Zertifikaten oder Anforderungen verwendet werden können.
- Kurz: X.509 graphisch und nachvollziehbar ;–)
- Webseite: https://www.hohnstaedt.de/xca/ [https://www.hohnstaedt.de/xca/]

2.5. SOFTWARE DOWNLOADS PRÜFEN

2.5.1. Debian CD/DVD Download

- Prüfen, ob eine Debian Installations–CD/DVD "original" ist, d.h. vom Debian–Release Team stammt:
- CD–Rom ISO Image laden

```
wget https://debian.inf.tu-dresden.de/debian-cd/13.1.0/amd64/iso-cd/debian-13.1.0-amd64-netinst.iso
```

- SHA256 Fingerabdruck der Datei errechnen

```
openssl dgst -sha256 < debian-13.1.0-amd64-netinst.iso
```

- Datei mit den Hash–Prüfsummen laden

```
wget https://debian.inf.tu-dresden.de/debian-cd/13.1.0/amd64/iso-cd/SHA256SUMS
```

- Hash der ISO–Datei mit dem Hash in der Prüfsummendatei vergleichen
- Nun prüfen wir, ob die Datei mit den Hash–Prüfsummen original ist. Dabei wird die digitale Signatur (mittels GnuPG) über die Datei mit den SHA256–Hashes geprüft. Wir laden die Signatur der Hash–Prüfsummen vom Download–Server:

```
wget https://debian.inf.tu-dresden.de/debian-cd/13.1.0/amd64/iso-cd/SHA256SUMS.sign
```

- Releases der Debian–Distribution werden mit extra Schlüsseln des Debian–Release–Teams signiert. Wenn wir direkt versuchen, die Signatur auf der Datei zu prüfen, so bekommen wir einen Fehler, da wir den öffentlichen Schlüssel (ID DF9B9C49EAA9298432589D76DA87E80D6294BE9B) des Debian–Release–Teams nicht in unserem GPG–Schlüsselbund haben:

```
debian$ gpg --verify SHA256SUMS.sign SHA256SUMS
gpg: directory '/home/user/.gnupg' created
gpg: keybox '/home/user/.gnupg/pubring.kbx' created
gpg: Signature made Sat Sep  6 21:54:28 2025 UTC
gpg:                using RSA key DF9B9C49EAA9298432589D76DA87E80D6294BE9B
gpg: Can't check signature: No public key
```

- Um die Signaturen prüfen zu können, importieren wir den Debian CD Signatur–Schlüssel in den eigenen GPG–Schlüsselring.

```
# sudo apt -y install dirmngr
# gpg  --keyserver keyring.debian.org --recv-keys
DF9B9C49EAA9298432589D76DA87E80D6294BE9B
gpg: key DA87E80D6294BE9B: public key "Debian CD signing key <debian-
cd@lists.debian.org>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

- Fingerabdruck des Schlüssels anzeigen und mit den Daten auf https://www.debian.org/CD/verify.html [https://www.debian.org/CD/verify.html] abgleichen

```
gpg --finger  DF9B9C49EAA9298432589D76DA87E80D6294BE9B
```

- Signature auf der Prüfsummendatei prüfen

```
# gpg --verify SHA256SUMS.sign SHA256SUMS
gpg: Signature made Sat Sep  6 21:54:28 2025 UTC
gpg:                using RSA key DF9B9C49EAA9298432589D76DA87E80D6294BE9B
gpg: Good signature from "Debian CD signing key <debian-cd@lists.debian.org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: DF9B 9C49 EAA9 2984 3258  9D76 DA87 E80D 6294 BE9B
```

- Um dem CD-ROM Image vertrauen zu können, sollten alle Prüfsummen und Signaturen stimmen.

2.5.2. Debian Release/Repository Schlüssel

- Die Debian-GPG Schlüssel zum verifizieren von Software-Paketen liegen im Verzeichnis `/etc/apt/trusted.gpg.d/`

2.5.3. Debian Release Informationen

- https://wiki.debian.org/SecureApt [https://wiki.debian.org/SecureApt]
- https://keyring.debian.org/ [https://keyring.debian.org/]
- https://ftp-master.debian.org/keys.html [https://ftp-master.debian.org/keys.html]
- https://launchpad.net/debian/+source/debian-archive-keyring/+changelog [https://launchpad.net/debian/+source/debian-archive-keyring/+changelog]
- https://debian-handbook.info/browse/stable/sect.package-authentication.html [https://debian-handbook.info/browse/stable/sect.package-authentication.html]

2.5.4. Die Authentizität und Integrität eines Debian Pakets manuell prüfen

- Debian Pakete sind über eine Signatur-Kette geschützt (anders als RPM Pakete bei RedHat/CentOS/Fedora, welche direkt durch eine GPG-Signatur geschützt sind)
- Die Hash-Prüfsummen der Debian-Pakete sind in den Paketlisten-Dateien hinterlegt (z.B. `main/Contents-amd64`). Die Hash-Prüfsummen der Paketlisten sind in der `Release` Datei hinterlegt, und diese Datei ist mit den GPG-Schlüssel des Debian-Release-Teams unterschrieben.
- Um nun ein beliebiges Debian-Paket zu prüfen
  - Errechne die SHA256 Prüfsumme des Pakets
  - Vergleiche diese Prüfsumme mit der Prüfsumme in der Paketliste der Architektur
  - Errechne die SHA256-Prüfsumme der Paketlisten Datei.

```
    wget https://debian.inf.tu-dresden.de/debian/dists/Debian13.1/main/binary-
amd64/Packages.gz
    openssl dgst -sha256 < Packages.gz
    zcat Packages.gz | less
```

- Lade die aktuelle `Release`-Datei für die Debian-Version:

```
$ wget https://debian.inf.tu-dresden.de/debian/dists/Debian13.1/Release
$ wget https://debian.inf.tu-dresden.de/debian/dists/Debian13.1/Release.gpg
```

- Importiere die (öffentlichen) GPG-Schlüssel des Debian-Release Teams. Hat man keine (vertrauenswürdige Debian-Installation), so findet man die Schlüssel unter https://ftp-master.debian.org/keys.html [https://ftp-master.debian.org/keys.html]

```
$ gpg --import /etc/apt/trusted.gpg.d/debian-archive-*
```

- Als letzter Schritt werden die Signaturen auf der `Release`-Datei geprüft

```
gpg --verify Release.gpg Release
```

- Alle Schritte dieser Prüfung müssen erfolgreich abgeschlossen sein, um der Installationsdatei (DEB-Paket) vertrauen zu können. Diese Prüfung wird von den Debian-Paketmanager-Programmen (`apt`, `apt-get`, `aptitude`, `debootstrap`) automatisch durchgeführt.

2.5.5. Neues APT-Repository mit GPG-Schlüssel hinzufügen

- Am Beispiel des offiziellen NGINX Repositories (neue NGINX Versionen, welche nicht in der Basis-Debian Distro enthalten sind)
- Lange Key-ID == Fingerprint des Schlüssels
- Key-ID mit Installations-Informationen auf der (TLS-gesicherten) NGINX Webseite abgleichen (https://nginx.org/en/linux_packages.html#Debian [https://nginx.org/en/linux_packages.html#Debian])

```
# sudo -s
# apt update
# apt -y install apt-transport-https ca-certificates
# curl -fsSL https://nginx.org/keys/nginx_signing.key > /etc/apt/trusted.gpg.d/nginx-
signing-key.asc
# gpg --show-keys --with-fingerprint /etc/apt/trusted.gpg.d/nginx-signing-key.asc
pub   rsa4096 2024-05-29 [SC]
      8540 A6F1 8833 A80E 9C16  53A4 2FD2 1310 B49F 6B46
uid                      nginx signing key <signing-key-2@nginx.com>

pub   rsa2048 2011-08-19 [SC] [expires: 2027-05-24]
      573B FD6B 3D8F BC64 1079  A6AB ABF5 BD82 7BD9 BF62
uid                      nginx signing key <signing-key@nginx.com>

pub   rsa4096 2024-05-29 [SC]
      9E9B E90E ACBC DE69 FE9B  204C BCDC D8A3 8D88 A2B3
uid                      nginx signing key <signing-key-3@nginx.com>
```

- Es sollte der volle Fingerabdruck der Schlüssels ausgegeben werden (mit der NGINX Webseite vergleichen https://nginx.org/en/pgp_keys.html [https://nginx.org/en/pgp_keys.html]).
- NGINX Repository hinzufügen

```
# echo "deb http://nginx.org/packages/debian trixie nginx" >
/etc/apt/sources.list.d/nginx.list
```

- NGINX installieren und starten

```
# apt update
Hit:1 http://nginx.org/packages/debian trixie InRelease
[...]
# apt install nginx
Installing:
  nginx

Installing dependencies:
  logrotate

Summary:
  Upgrading: 0, Installing: 2, Removing: 0, Not Upgrading: 0
  Download size: 1104 kB
  Space needed: 3797 kB / 6419 MB available
  Continue? [Y/n]
Get:1 http://nginx.org/packages/debian trixie/nginx amd64 nginx amd64 1.28.0-1~trixie
[1042 kB]
Get:2 file:/etc/apt/mirrors/debian.list Mirrorlist [39 B]
Get:3 http://mirrors.digitalocean.com/debian trixie/main amd64 logrotate amd64 3.22.0-1
[62.0 kB]
Fetched 1104 kB in 0s (9442 kB/s)
Selecting previously unselected package logrotate.
(Reading database ... 49651 files and directories currently installed.)
Preparing to unpack .../logrotate_3.22.0-1_amd64.deb ...
Unpacking logrotate (3.22.0-1) ...
Selecting previously unselected package nginx.
Preparing to unpack .../nginx_1.28.0-1~trixie_amd64.deb ...
----------------------------------------------------------------------

Thanks for using nginx!

Please find the official documentation for nginx here:
 * https://nginx.org/en/docs/

Please subscribe to nginx-announce mailing list to get
the most important news about nginx:
 * https://nginx.org/en/support.html

Commercial subscriptions for nginx are available on:
 * https://nginx.com/products/

----------------------------------------------------------------------
Unpacking nginx (1.28.0-1~trixie) ...
Setting up logrotate (3.22.0-1) ...
Created symlink '/etc/systemd/system/timers.target.wants/logrotate.timer' →
'/usr/lib/systemd/system/logrotate.ti
mer'.
logrotate.service is a disabled or a static unit, not starting it.
Setting up nginx (1.28.0-1~trixie) ...
Created symlink '/etc/systemd/system/multi-user.target.wants/nginx.service' →
'/usr/lib/systemd/system/nginx.serv
ice'.
Processing triggers for man-db (2.13.1-1) ...
```

```
# systemctl start nginx
# systemctl status nginx
```

2.6. MINIMALE INSTALLATIONEN

- Eine Minimale Linux–Betriebssysteminstallation verringert die Angriffsfläche auf das System und verringert die Komplexität (weniger Fehlerquellen)
- Virtuelle Maschinen oder Linux–Container erlauben es, dedizierte Linux–Installationen zu betreiben mit einer minimalen Anzahl von installierten Paketen
- Pakete welche in der Regel nicht auf sicherheits–sensitiven Servern installiert werden sollten
  - Grafische Benutzeroberflächen (X11, Wayland, Gnome, KDE etc)
  - Entwicklungstools (Compiler, Skript–Sprachen, Linker, Debugger …)
  - Tools welche für Angriffe auf andere Systeme benutzt werden können (nmap, nc, scapy …)
  - Komplexe Softwarepakete (LibreOffice, Emacs, LaTeX …)
- Minimale Debian–Installationen können mit dem Programm `debootstrap` in ein Verzeichnis erstellt werden
  - Dieses Verzeichnis kann als Ausgangssystem für ein Linux–Container benutzt werden
  - Das Verzeichnis kann über ein Linux–Live–System (Rettungssystem) auf eine virtuelle Maschine oder einen physischen Server aufgespielt werden
- Erstelle ein leeres Verzeichnis für ein minimales Debian–System

```
mkdir -p /srv/container/debian
```

- Installiere das `debootstrap` Programm

```
# apt install debootstrap
```

- Installiere eine Debian 12 (bookworm) Distribution in das Verzeichnis `/srv/container/debian`

```
debootstrap --arch=amd64 bookworm /srv/container/debian
```

- Größe der minimales Debian–Installation (ca. 300 MB)

```
# du -sh /srv/container/*
```

- Installiere Systemd–Container–Tools

```
apt install systemd-container
```

- Starte eine Shell innerhalb von Linux–Namespaces mittels `systemd-nspawn`

```
systemd-nspawn -D  /srv/container/debian
```

- Root–Passwort setzen und einen neuen Benutzer erstellen

```
root@debian$ passwd
root@debian$ adduser user
```

- Die Namespaces verlassen

```
root@debian# exit
```

- Debian-System in einen `systemd-nspawn` Container starten

```
systemd-nspawn -bD  /srv/container/debian
```

- Mit dem Befehl `poweroff` kann der Container ausgeschaltet werden

```
root@debian$ poweroff
```

2.7. LINUX-DISTRIBUTIONEN (SICHERHEITSBEWERTUNG)

- Debian 12/13
  - MAC ++ (AppArmor, SELinux, TOMOYO)
  - BuildHardening ++ (reproducible Builds)
  - Geschwindigkeit von Sicherheits-Patches:
  - Minimale Installation:
- Ubuntu – https://wiki.ubuntu.com/Security/Features [https://wiki.ubuntu.com/Security/Features]
  - MAC: ++ (AppArmor, SELinux)
  - Build Hardening ++
  - Geschwindigkeit von Sicherheits-Patches: ++
  - Minimale Installation: O
- SUSE – https://en.opensuse.org/openSUSE:Security_Features [https://en.opensuse.org/openSUSE:Security_Features]
  - MAC: ++ (AppArmor, SELinux)
  - Build Hardening:
  - Geschwindigkeit von Sicherheits-Patches: ++
  - Minimale Installation: –
- Red Hat / AlmaLinux / RockyLinux / Fedora – https://fedoraproject.org/wiki/Security_Features_Matrix [https://fedoraproject.org/wiki/Security_Features_Matrix]
  - MAC: + (SELinux)
  - Build Hardening:
  - Geschwindigkeit von Sicherheits-Patches:
  - Minimale Installation: –
- Gentoo (Hardened) – https://wiki.gentoo.org/wiki/Hardened_Gentoo/de [https://wiki.gentoo.org/wiki/Hardened_Gentoo/de]
  - MAC: ++ (SELinux, AppArmor, TOMOYO)

- Build Hardening: ++
- Geschwindigkeit von Sicherheits-Patches: O
- Minimale Installation: ++

2.8. SOFTWARE-SICHERHEITSFUNKTIONEN UNTER LINUX

2.8.1. Kernel-Konfigurationsoptionen

Der Linux-Kernel bietet einige Sicherheitfunktionen, welche zur Zeit der Übersetzung des Kernels aus den Quellen angeschaltet werden können. Einige dieser Funktionen haben negative Auswirkungen auf die Ausführungsgeschwindigkeit des Systems, so das diese Funktionen nicht in allen Distributionen gesetzt sind. In den Klammern sind die jeweiligen Konfigurationsnamen aufgeführt. In einem Linux-System können diese Namen in der Kernel-Konfigurationsdatei geprüft werden. Beispiel:

```
zcat /proc/config.gz | grep CONFIG_SECURITY_DMESG_RESTRICT
```

oder, wenn `/proc/config.gz` nicht vorhanden.

```
cat  /boot/config-$(uname -r) | grep CONFIG_SECURITY_DMESG_RESTRICT
```

Die Funktionen (Kernel 4.15+):

- Zugriff auf Kernel-Syslogmeldungen via dmesg unterbinden (CONFIG_SECURITY_DMESG_RESTRICT)
- Speicher-Kopierroutinen zwischen Kernel- und Userspace härten (CONFIG_HARDENED_USERCOPY)
- String- und Speicher-Funktionen gegen Buffer-Overflows härten (CONFIG_FORTIFY_SOURCE)
- LoadPIN-Support: Option um alle Kernel-Dateien auf ein (nur-lese) Dateisystem (Module, Firmware, LSM-Dateien) zu beschränken (CONFIG_SECURITY_LOADPIN)

Kernel-Schnittstelle für Hardware- und CPU Sicherheitsprobleme

- Seit Kernel 4.15 besitzt der Linux-Kernel eine Schnittstelle zum Abfragen von bekannten Hardware- und CPU-Sicherheitsproblemen, und Informationen ob der laufende Linux-Kernel gegen die Sicherheitsprobleme schützt.

```
grep . /sys/devices/system/cpu/vulnerabilities/*
```

2.8.2. Userspace Software-Sicherheitsfunktionen unter Linux

- RELRO [http://tk-blog.blogspot.de/2009/02/relro-not-so-well-known] – RELocation Read-Only
- PIE [http://blog.fpmurphy.com/2008/06/position-independent-executables.html] – Position Independent Executable
- ASLR [https://de.wikipedia.org/wiki/Address_Space_Layout_Randomization] – Address Space Layout Randomization
- Fortify Source [https://access.redhat.com/blogs/766093/posts/1976213] – Array Bounds-Checks etc.

- Stack protector/StackCanary [https://xorl.wordpress.com/2010/10/14/linux-glibc-stack-canary-values/] – Stack Canary
- NX-No-Execute [https://en.wikipedia.org/wiki/NX_bit] – Daten im Speicher als *nicht ausführbar* (NX=no execute) markieren
- Hintergrund-Informationen zu diesen Sicherheitsfunktionen: Notes on Build Hardening (December 15, 2018) https://blog.erratasec.com/2018/12/notes-on-build-hardening.html [https://blog.erratasec.com/2018/12/notes-on-build-hardening.html]
- Studie von Sicherheitsfunktionen in Linux-Distributionen: Millions of Binaries Later: a Look Into Linux Hardening in the Wild (February 28, 2019) https://web.archive.org/web/20190302014002/https://capsule8.com/blog/millions-of-binaries-later-a-look-into-linux-hardening-in-the-wild/ [https://web.archive.org/web/20190302014002/https://capsule8.com/blog/millions-of-binaries-later-a-look-into-linux-hardening-in-the-wild/]

Programm zum Prüfen der Sicherheitsfunktionen

- Github Repository https://github.com/slimm609/checksec.sh [https://github.com/slimm609/checksec.sh]

```
apt install git binutils
git clone https://github.com/slimm609/checksec.sh
cd checksec.sh/
bash ./checksec.bash --proc-all
bash ./checksec.bash --kernel
bash ./checksec.bash --dir=/usr/sbin --verbose
```

2.9. KERNEL PARAMETER

- Kernel Parameter können auf der Kernel-Kommando-Zeile im Bootloader angegeben werden, um Kernel-Funktionen zu steuern

2.9.1. Audit-Subsystem

- Audit-Subsystem anschalten

```
audit=[0|1]
```

- wenn `audit` nicht gesetzt ist, dann wir das Audit-Subsystem erst mit dem Starten des Audit-Daemon `auditd` aktiv
- Wert 0 — Audit-Subsystem ist ausgeschaltet
- Wert 1 — Audit-Subsystem wird sofort aktiv, der Kernel sammelt Audit Informationen und übergibt diese an den Audit-Daemon, sobald dieser gestartet ist

2.9.2. AppArmor

```
apparmor=[0|1]
```

- AppArmor an/ausschalten

2.9.3. SELinux

- SELinux an/ausschalten

```
selinux=[0|1]
```

- SELinux Regeln durchsetzen (enforcing)

```
enforcing=[0|1]
```

- Wert 0 = SELinux im permissive Modus
- Wert 1 = SELinux im enforcing Modus

2.9.4. Signierte Module

```
module.sig_enforce
```

- wenn gesetzt, können nur Kernel–Module mit einer gültigen Signatur geladen werden

2.9.5. NOEXEC

```
noexec=[on|off]
```

- bei 32bit PAE Kerneln, schaltet Schreibschutz auf Daten–Speicherseiten an

2.9.6. Datei–Capabilities

```
no_file_caps
```

- Schaltet die Datei–Capabilities aus

2.9.7. Keine Module nachladen

```
nomodule
```

- es können keine Module nachgeladen werden

2.9.8. Kernel–Panic

```
panic=<timeout>
```

- Wert > 0 – Sekunden bis zum Reboot
- Wert = 0 – kein Reboot (* aus Sicherheitsgründen empfohlen)
- Wert < 0 – sofortiger Reboot

2.9.9. sysctl Kernel-Variablen

```
sysctl -a
```

- **Permanente** `sysctl` **Einstellungen in** `/etc/sysctl.conf`

```
kernel.randomize_va_space = 2

# Restrict core dumps
fs.suid_dumpable = 0

# Hide exposed kernel pointers
kernel.kptr_restrict = 1

#Prevent SYN attack, enable SYNcookies (they will kick-in when the max_syn_backlog
reached)
# use iptables synproxy
net.ipv4.tcp_syncookies = 1
net.ipv4.tcp_syn_retries = 2
net.ipv4.tcp_synack_retries = 2
net.ipv4.tcp_max_syn_backlog = 4096

# Disables packet forwarding
net.ipv4.ip_forward = 0
net.ipv4.conf.all.forwarding = 0
net.ipv4.conf.default.forwarding = 0
net.ipv6.conf.all.forwarding = 0
net.ipv6.conf.default.forwarding = 0

# Disables IP source routing
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0

# Enable IP spoofing protection, turn on source route verification
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

# Disable ICMP Redirect Acceptance
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0

# Enable Log Spoofed Packets, Source Routed Packets, Redirect Packets
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1

# Don't relay bootp
net.ipv4.conf.all.bootp_relay = 0
```

```
# Don't proxy arp for anyone
net.ipv4.conf.all.proxy_arp = 0

# Enable ignoring broadcasts request
net.ipv4.icmp_echo_ignore_broadcasts = 1

# Enable bad error message Protection
net.ipv4.icmp_ignore_bogus_error_responses = 1
```

2.10. LINUX SICHERHEITSMELDUNGEN

Quelle	Link
Linux Weekly News Sicherheitsmeldungen	https://lwn.net/Security/ [https://lwn.net/Security/]
Red Hat Sicherheitsmeldungen	https://access.redhat.com/security/security-updates/#/security-advisories
Debian Security Tracker	https://security-tracker.debian.org/tracker/ [https://security-tracker.debian.org/tracker/]
Debian "stable" Sicherheitsmeldungen	https://security-tracker.debian.org/tracker/status/release/stable [https://security-tracker.debian.org/tracker/status/release/stable]
SUSE Sicherheitsmeldungen	https://www.suse.com/de-de/support/update/ [https://www.suse.com/de-de/support/update/]
Ubuntu Sicherheitsmeldungen	https://www.ubuntu.com/usn/ [https://www.ubuntu.com/usn/]
Gentoo Sicherheitsmeldungen	https://security.gentoo.org/glsa [https://security.gentoo.org/glsa]
BSI Technische Sicherheitshinweise und Warnungen	https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Cyber-Sicherheitslage/Technische-Sicherheitshinweise-und-Warnungen/technische-sicherheitshinweise-und-warnungen_node.html [https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Cyber-Sicherheitslage/Technische-Sicherheitshinweise-und-Warnungen/technische-sicherheitshinweise-und-warnungen_node.html]

2.11. SICHERHEIT BEI DER UNIX-BENUTZERANMELDUNG

- Die Sicherheit der Unix-Benutzeranmeldung hängt u.a. an der Sicherheit der gespeicherten Passwörter. Passwörter werden in einem Linux-System als SHA512- oder Yescrypt-Hash gespeichert. Dabei wird das Passwort 5.000 mal mit dem Algorithmus gehashed bevor es in der Datei `/etc/shadow` gespeichert wird oder gegen den Hash in dieser Datei geprüft wird.
- 5.000 Runden lassen sich heute mit schnellen (Cloud-)Rechnern "brute-force" berechnen. Die Default-Einstellung von Linux ist so gewählt, das es auch auf sehr schwachen Rechnern (z.B. Heim-Routern) noch funktioniert. Moderne Systeme können eine grössere Anzahl Hash-Runden für die Passwort-Sicherheit verwenden. Bei modernen Server-Rechnern kann bei SHA512-Passwort-Hashing die Anzahl der Runden auf 1.000.000 (1 Million) eingestellt werden, ohne das eine merkliche Verzögerung des Anmeldeprozesses auftritt. Ähnlich kann die Stärke des Yescrypt Hashing angepasst werden.

2.11.1. Sicherheit der Benutzerpasswörter

- In der Datei `/etc/pam.d/common-password` wird die Stärke der Benutzer-Passwörter angepasst. Eine Änderung in dieser Datei wirkt sich nur auf alle neu gesetzten Passwörter aus, alle schon vorher vergebenen Passwörter bleiben mit der vorherigen Einstellung bestehen. D.h. nach einer Änderung dieses Parameters sollten wichtige Passwörter neu vergeben werden.

```
[...]
password    [success=1 default=ignore]    pam_unix.so obscure sha512 rounds=1000000
[...]
```

2.11.2. Yesycrypt ab Debian 11

- *Yescrypt* ist eine neue Schlüsselableitungsfunktion für Passwörter vom OpenWall Projekt. Es basiert auf scrypt [https://www.tarsnap.com/scrypt.html] (RFC 7914 [https://datatracker.ietf.org/doc/html/rfc7914]) von Colin Percival (früherer Sicherheitleiter des FreeBSD Projektes). Es schützt besser vor Brute-Force Angriffen als SHA512 und andere Schlüsselableitungsfunktionen.
- `yescrypt` mit der Spezifikation `yescrypt v2` ist weit verbreitet und das Standard-Passwort-Hashing-Schema für aktuelle Versionen der wichtigsten Distributionen wie Debian 11+, Fedora 35+, RHEL 9+, Kali Linux 2021.1+ und Ubuntu 22.04+. Dennoch unterstützen viele Standard-Tools `yescrypt` noch nicht.
  - Beim Update von Linux Distributionen werden die Passwörter nicht neu mit den besseren Algorithmen gehashed (das geht technisch nicht, da das Original-Passwort nicht vorliegt). Um die Vorteile der neuen Passwort-Hashing-Algorithmen wie `yescrypt` benutzen zu können, müssen die Passwörter neu gesetzt werden.
    - erst bei einer Passwortänderung mittels `passwd` wir der neue Yescrypt-Hash erzeugt
  - Beispiel Debian 10 SHA512 Hash in der `/etc/shadow`

```
nutzer:$6$NtILOHbh$SZ1pFZKkJj/xqtrxqtSBrkOgRWQmXfv4tJhwlCfNkL7V4ft8G6eyLkVhxmvAs6DQsPuA
mRqB7WVfJHMB5w0340:16442:0:99999:7:::
```

- Beispiel Debian 11 Yescrypt Hash in der `/etc/shadow`

```
nutzer:$y$j9T$YhVeKuFhMSSA91rR4FhwT/$9YTuWLhjyf1oqcPMoEkJFqSL.6YMRMQVMGaIgWIJyq9:18868:
```

```
0:99999:7:::
```

- Yescrypt in Debian 11 https://www.debian.org/releases/bullseye/amd64/release-notes/ch-information.en.html#pam-default-password [https://www.debian.org/releases/bullseye/amd64/release-notes/ch-information.en.html#pam-default-password]
- Yescrypt Informationen https://www.openwall.com/yescrypt/ [https://www.openwall.com/yescrypt/]
- `yescrypt` Dokumentation: https://github.com/openwall/yescrypt/blob/main/README [https://github.com/openwall/yescrypt/blob/main/README]

pam_unix und yescrypt

- Yescrypt benutzt einen cost factor [1..11] anstelle der rounds

```
[...]
password    [success=1 default=ignore]    pam_unix.so obscure yescrypt rounds=10
[...]
```

Man erkennt es am Parameterteil im gehashten passwort (shadow) `$jDT` statt `$j9T` Standard sind 5 "Runden"

2.11.3. Gruppenpasswörter

- Die Sicherheit der Gruppenpasswörter werden in der Datei `/etc/login.defs` festgelegt. Wenn Gruppenpasswörter benutzt werden, wird empfolen diese Werte mit der PAM-Konfiguration gleich zu halten.

```
ENCRYPT_METHOD SHA512
SHA_CRYPT_MIN_ROUNDS 1000000
SHA_CRYPT_MAX_ROUNDS 1000000
```

2.11.4. Benutzerdatenbank

- Die Benutzerinformationen für die Passwortanmeldung unter Unix/Linux werden in der Datei `/etc/shadow` gespeichert

Felder der Datei `/etc/shadow`:

- Benutzername
- Password-Hash
- Datum des letzen Passwort-Wechsel
- Mindestlebensdauer des Passworts
- Max-Lebensdauer des Passwort
- Passwort-Ablauf Warn-Zeitraum
- Zeitdauer der Passwort-Inaktivität (Benutzer kann sich nach Ablauf des Passworts noch einloggen)
- Ablaufdatum des Passworts

- Reserviertes Feld

Passwort-Hash-Methoden

ID	Methode
1	MD5
2a	Blowfish (nicht in der Standard glibc; wurde einigen Distributionen hinzugefügt)
5	SHA-256 (seit glibc 2.7)
6	SHA-512 (seit glibc 2.7)
y/7	yescrypt (Debian 11+/RHEL 9+/Fedora 35+/Ubuntu 22.04+)

2.12. CHAGE

- Der Linux-Befehl `chage` wird benutzt um das Ablaufdatum eines Passwortes zu verwalten
- Benutzer anlegen, Passwort vergeben und aktuelle Passwort-Lebenszeit ausgeben

```
# useradd -m -N fritz
# passwd fritz
# chage -l fritz
Last password change                                    : Sep 29, 2025
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

- Account als abgelaufen markieren

```
chage -E 0 fritz
```

- Account ist ab einem bestimmten Datum nicht mehr gültig

```
chage -E 2019-03-01 fritz
```

- Ablaufzeit zurücksetzen

```
chage -E -1 fritz
```

- Passwort darf frühstens nach 3 und muss spätestens nach 9 Tagen geändert werden

```
chage -m 3 -M 9 -d "1 day ago" fritz
passwd fritz
```

- Login innerhalb der Warn–Periode

```
chage -d "8 days ago" -W 3 fritz
su - fritz
```

- Login außerhalb der Warn–Periode aber innerhalb der Gültigkeit

```
chage -d "12 days ago" -I 10 fritz
su - fritz
```

- Login außerhalb der Gültigkeit

```
chage -d "21 days ago" -I 10 fritz
su - fritz
```

- Benutzer muss sein Passwort einmalig beim ersten Login ändern

```
chage -E -1 -I -1 -m 0 -M 99999 fritz # alles zurücksetzen
passwd fritz                          # Ein Standardpasswort setzen
```

- Passwortänderung nach der initialen Anmeldung erzwingen

```
chage -d 0 fritz
```

2.12.1. Links zur Passwort Sicherheit

- Auch das BSI verabschiedet sich vom präventivem,regelmäßigen Passwort–Wechsel: https://www.heise.de/news/Passwoerter–BSI–verabschiedet–sich–vom–praeventiven–Passwort–Wechsel–4652481.html [https://www.heise.de/news/Passwoerter–BSI–verabschiedet–sich–vom–praeventiven–Passwort–Wechsel–4652481.html]

2.13. SU

- mit dem Programm `su` (Switch User eigentlich Substitute User) kann ein Benutzer ein einen anderen Benutzerkontext wechseln
- Unterschiede bei den Umgebungsvariablen zwischen `su` und `su -` oder `su -l`
  - bei `su` werden die Umgebung des aufrufenden Benutzer übernommen (kann Sicherheitsprobleme erzeugen)
  - die Variablen `$IFS`, `$HOME`, `$SHELL`, `$USER`, `$LOGNAME`, `$PATH` werden bei `su -` oder `su -l` zurückgesetzt
- Such–Pfade für Benutzer und `root` werden in `/etc/login.defs` über die Optionen `ENV_PATH` und `ENV_SUPATH` konfiguriert

2.14. SUDO

- `sudo` ist ein moderner Ersatz für `su`. Gegenüber `su` hat `sudo` mehrere Vorteile:
  - Es wird das eigene Benutzerpasswort abgefragt, nicht das Passwort dies Ziel–Benutzers
  - Das Ergebniss der Passwort–Prüfung des Benutzers kann für eine gewisse Zeit gespeichert werden, so das der Benutzer nicht für jeden Befehl das Passwort eingeben muss

    - Bessere Protokollierung
    - Umfangreiche Konfigurationsmöglichkeiten
- `sudo` Installation

```
apt install sudo
```

- `sudo` Basis-Konfiguration ausgeben

```
sudo -V
```

- `sudo` Konfigurationsdatei sicher editieren (ggf. Variable `$EDITOR` setzen, sonst wird `vi` verwendet)

```
EDITOR=emacs visudo
```

2.14.1. Beispiel:

- Befehl `cat` auf die Datei `/var/log/apt/term.log` für Benutzer `user`

```
visudo -f /etc/sudoers.d/apt-term-cat
----
user ALL=(root) /bin/cat /var/log/apt/term.log
----
user$ sudo -l
```

- `sudo -l` Zeigt die `sudo`-Konfiguration und die erlaubten Befehle eines `sudo`-Benutzers an.

2.14.2. Aufgabe:

- Erstelle eine `sudo` Konfiguration, um es dem Benutzer **user** zu erlauben, die Logdatei `/var/log/dpkg.log` unter den Benutzerberechtigungen des Benutzers `root` mit den Programm `less` ohne Eingabe eines Passworts anzuschauen (Dokumentation der Datei `sudoers` anschauen mit `man sudoers`)
- Der folgende Befehl sollte als Benutzer `user` ohne Eingabe eines Passworts funktionieren

```
    sudo less /var/log/dpkg.log
```

2.14.3. Lösung

```
visudo -f /etc/sudoers.d/log-message-view
------
...
user  ALL=(root) NOPASSWD: /usr/bin/less /var/log/dpkg.log
...
```

- in einem anderen Terminal/Tmux-Fenster, teste **sudo** als Benutzer **user**

```
user$ sudo less /var/log/dpkg.log # <--- sollte ohne Eingabe des Passwort angezeigt
werden
user$ sudo less /etc/shadow       # <--- darf nicht ohne Eingabe des Passwort
angezeigt werden
```

2.14.4. Frage:

- gibt es mit dieser Konfiguration ein Sicherheitsproblem?

2.14.5. Antwort:

- Ja – `less` und viele andere Programme können Unterprogramme aufrufen (z.B. eine Shell), welche dann mit root–Rechten läuft
- Lösung: NOEXEC:

```
user  ALL=(root) NOEXEC: /usr/bin/more /var/log/messages
user  ALL=(root) NOEXEC: /usr/bin/more /var/log/boot.log
```

2.14.6. sudoedit

- Benutzer user soll die Datei `/etc/rsyslog.conf` editieren dürfen
- in der Datei `/etc/sudoers`

```
visudo
----
user ALL= sudoedit /etc/rsyslog.conf
----
sudoedit /etc/rsyslog.conf
```

- Eine Gruppe in `sudo` berechtigen

```
visudo -f /etc/sudoers.d/admins
%sudo  ALL=(ALL) NOPASSWD: ALL
```

- Priviligierte Shells–Sessions mit `sudo`

```
sudo -s # entspricht su
sudo -i # entspricht su -
```

2.14.7. sudo Aliases

```
    # User alias specification
    User_Alias      FULLTIMERS = millert, mikef, dowdy
    User_Alias      PARTTIMERS = bostley, jwfox, crawl
    User_Alias      WEBMASTERS = will, wendy, wim

    # Runas alias specification
    Runas_Alias     OP = root, operator
    Runas_Alias     DB = oracle, sybase
    Runas_Alias     ADMINGRP = adm, oper

    # Host alias specification
    Host_Alias      SPARC = bigtime, eclipse, moet, anchor :\
                    LINUX = grolsch, dandelion, black :\
                    LINUX_ARM = widget, thalamus, foobar :\
                    LINUX_PPC64 = boa, nag, python
    Host_Alias      CUNETS = 128.138.0.0/255.255.0.0
```

```
    Host_Alias      CSNETS = 128.138.243.0, 128.138.204.0/24, 128.138.242.0
    Host_Alias      SERVERS = master, mail, www, ns
    Host_Alias      CDROM = orion, perseus, hercules

    # Cmnd alias specification
    Cmnd_Alias      KILL = /usr/bin/kill, /usr/bin/pkill
    Cmnd_Alias      PRINTING = /usr/sbin/lpc, /usr/bin/lprm
    Cmnd_Alias      SHUTDOWN = /usr/sbin/shutdown
    Cmnd_Alias      HALT = /usr/sbin/halt
    Cmnd_Alias      REBOOT = /usr/sbin/reboot
    Cmnd_Alias      SHELLS = /usr/bin/sh, /usr/bin/zsh, /bin/bash

FULLTIMERS  SPARC=(OP) KILL
WEBMASTERS  LINUX=(:ADMINGRP) SHELLS
```

2.14.8. sudo replay

- Füge die folgenden Konfigurationszeilen in die `/etc/sudoers` Datei ein

```
Defaults log_output
Defaults!/usr/bin/sudoreplay !log_output
Defaults!/sbin/reboot !log_output
```

- user darf root werden

```
user ALL=(root) ALL
```

- Benutze `sudo` als Benutzer user um eine interaktive Shell zu bekommen

```
user$ sudo -i
```

- Ein paar Befehle ausführen, die Ausgaben produzieren
- Die `sudo` Root-Shell wieder verlassen
- (im Terminal als Benutzer root) Aufgezeichnete Sitzungen auflisten

```
sudoreplay -l
```

- Aufgezeichnete `sudo` Sitzung abspielen

```
sudoreplay <TSID>
```

- Verzeichnis der `sudo` Aufzeichnungen:

```
ls -l /var/log/sudo-io/
```

2.14.9. Einfache Intrusion Detection mit `sudo` (ab sudo 1.8.7)

```
openssl dgst -sha256 /usr/bin/passwd
SHA256(/usr/bin/passwd)=
a92b1b6fb52549ed23b12b32356c6a424d77bcf21bfcfbd32d48e12615785270
visudo
----
```

```
user ALL= sha256:a92b1b6fb52... /usr/bin/passwd
----
```

2.14.10. sudo Konfiguration (Passwort Cache Beispiele)

```
Defaults passwd_tries=5, passwd_timeout=2
Defaults timestamp_timeout=0 # Disable password caching
Defaults timestamp_timeout=5 # 5 Minuten password caching (default)
```

2.14.11. Links zu "sudo"

- Sudo for blue teams: how to control and log better https://www.sudo.ws/posts/2022/05/sudo-for-blue-teams-how-to-control-and-log-better/ [https://www.sudo.ws/posts/2022/05/sudo-for-blue-teams-how-to-control-and-log-better/]

2.15. DOAS AKA "DEDICATED OPENBSD APPLICATION SUBEXECUTOR"

- `doas` ist eine weniger komplexe Implementation der `sudo`-Idee aus dem OpenBSD Projekt. `doas` wurde erstellt, da den OpenBSD Entwicklern `sudo` zu komplex wurde und immer wieder Sicherheitprobleme mit `sudo` aufgetreten sind
- Webseite der *portablen* (nicht OpenBSD) `doas` Version: https://github.com/Duncaen/OpenDoas [https://github.com/Duncaen/OpenDoas]
- `doas` Installation

```
# apt install doas
```

- `doas` Konfigurationsdatei anlegen: `/etc/doas.conf`

```
# Sample file for doas
# Please see doas.conf manual page for information on setting
# up a doas.conf file.

# Permit members of the wheel group to perform actions as root.
permit :wheel

# Same without having to enter the password
permit nopass :wheel

# Permit user alice to run commands as a root user.
permit alice as root

# Permit user bob to run programs as root, maintaining
# environment variables. Useful for GUI applications.
permit keepenv bob as root

# Permit user cindy to run only the pkg package manager as root
# to perform package updates and upgrades.
permit cindy as root cmd pkg args update
permit cindy as root cmd pkg args upgrade

# Allow david to run id command as root without logging it
```

```
permit nolog david as root cmd id
```

2.15.1. Aufgabe zu doas

- Lese die Man-Page zu doas
- Erstelle eine doas.conf Konfiguration (eine Zeile reicht) welche …
  - … es dem Benutzer user erlaubt Programme unter dem Benutzer root auszuführen
  - … der Benutzer nach der Eingabe des Passworts für einige Zeit den Befehl doas ohne weitere Eingabe des Passworts benutzen kann

2.15.2. Lösung der Aufgabe zu doas

- Inhalt der Datei /etc/doas.conf

```
permit persist user as root
```

2.16. SYSTEMD SICHERHEIT

- https://gist.github.com/ageis/f5595e59b1cddb1513d1b425a323db04 [https://gist.github.com/ageis/f5595e59b1cddb1513d1b425a323db04]
- https://0pointer.de/blog/projects/security.html [https://0pointer.de/blog/projects/security.html]

2.16.1. Einfache Direktiven

- Units unter anderer uid/gid laufen lassen
- Zugriff auf Verzeichnisse beschränken
- Prozesslimits setzen
- Neue Systemd-Unit anlegen

```
systemctl edit --full --force simplehttp.service
```

- Inhalt der Systemd-Unit für einen einfachen Webserver auf Port 8080

```
[Unit]
Description=Python HTTP Server (do not use in production)

[Service]
Type=simple
Restart=on-failure
ExecStart=/usr/bin/python3 -m http.server 8080

# Namespace Spielereien
WorkingDirectory=/usr/share/doc
ReadOnlyDirectories=/proc /sys
InaccessibleDirectories=/var/log
PrivateTmp=yes

ProtectSystem=strict
```

```
ProtectHome=read-only

# Limits
# darf nicht forken
#LimitNPROC=1
# darf nicht ins fs schreiben
#LimitFSIZE=0

# laeuft als root. Capabilities
# entziehen
#CapabilityBoundingSet=~cap_chown,cap_net_raw
# setzen
#CapabilityBoundingSet=cap_net_raw
CapabilityBoundingSet=cap_net_bind_service

# laeuft als user. Capabilities
#User=gabi
#Group=users
DynamicUser=yes
AmbientCapabilities=cap_net_bind_service
```

- Erstelle die Systemd-Unit, starte den Dienst und teste den Webserver mit einem Browser auf http://serverXX.dane.onl:8080/ [http://serverXX.dane.onl:8080/]

2.16.2. Selbstanalyse

- Der Befehl `systemd-analyze security` listet alle im System definierten Systemd-Units mit einer Sicherheitsbewertung basierend auf den Beschränkungen der Service-Unit
- Der Befehl `systemd-analyze security <unit.service>` listet eine Übersicht der meisten sicherheitsrelevanten Systemd-Unit-Einstellungen einer Service-Unit

```
$ systemd-analyze security sshd
```

2.16.3. Weitere Directiven und Isolationstechniken in Systemd-Service-Units

- System-Härtung von Systemd-Units unter Debian 13 (und kompatiblen Systemen) im Abschnitt `[service]`
- Dokumentation: https://www.freedesktop.org/software/systemd/man/latest/systemd.exec.html [https://www.freedesktop.org/software/systemd/man/latest/systemd.exec.html]
- Nur 64bit Programme ausführen, 32bit i686 ist nicht erlaubt

```
Personality=x86-64
LockPersonality=yes
SystemCallArchitectures=x86-64
```

- Berechtigungsmaske für neue Dateien

```
UMask=077
```

- Das Setzen von SUID- und SGID-Bits auf Dateien ist nicht erlaubt

```
RestrictSUIDSGID=yes
```

- Maximaler Speicherverbrauch dieser Anwendung, muss ggf. an die Anwendung angepasst werden

```
MemoryMax=2G
```

- Maximale Anzahl von Prozessen/Threads in dieser Anwendung

```
TasksMax=20
```

- Anwendung darf keine Namespaces/Container erstellen

```
RestrictNamespaces=yes
```

- Anwendung darf die Echtzeit–Einstellungen nicht ändern

```
RestrictRealtime=yes
```

- Daten dürfen nicht ausgeführt werden (muss ggf. bei Java Just–in–Time Compiler ausgeschaltet werden)

```
MemoryDenyWriteExecute=yes
```

- Anwendung darf sich keine neuen Rechte verschaffen

```
NoNewPrivileges=yes
```

- Anwendung hat eine private Sicht auf die Gerätedateien unter `/dev`

```
PrivateDevices=yes
```

- Anwendung darf nur auf Standard–Geräte zugreifen

```
DevicePolicy=closed
```

- Anwendung bekommt ein privates `/tmp` Verzeichnis/Dateisystem

```
PrivateTmp=yes
```

- Prozess kann keine neuen Dateisysteme global mounten

```
PrivateMounts=yes
```

- Anwendung kann keine Änderungen an den CGroups vornehmen

```
ProtectControlGroups=yes
```

- Anwendung bekommt eine private Kopie der Heimverzeichnisse (`/home /root /run`)

```
ProtectHome=yes
```

- Anwendung kann keine Kernel–Module laden/entladen

```
   ProtectKernelModules=yes
```

- Anwendung kann keine Kernel-Parameter ändern

```
   ProtectKernelTunables=yes
```

- Anwendung kann keine Systemdateien ändern

```
   ProtectSystem=yes
```

- Prozess darf den Hostnamen nicht ändern

```
   ProtectHostname=yes
```

- Prozess bekommt einen leeren Network-Namespace und kann keine Netzwerkdaten (ausser privates Loopback) senden und empfangen

```
   PrivateNetwork=yes
```

- Anwendung kann nur Verbindungen mittels dieser Protokolle aufbauen

```
   RestrictAddressFamilies=AF_UNIX AF_INET AF_INET6
```

- Anwendung kann Syscalls aus diesen Anwendungsbereichen nicht benutzen

```
   SystemCallFilter=~@clock @debug @module @mount @raw-io @reboot @swap @cpu-emulation
@obsolete
```

- Anwendung bekommt die folgenden Capabilities entzogen

```
   CapabilityBoundingSet=~CAP_SETPCAP CAP_SYS_ADMIN CAP_SYS_PTRACE CAP_CHOWN CAP_FSETID
CAP_SETFCAP CAP_(DAC_OVERWRITE CAP_FOWNER CAP_IPC_OWNER
   CapabilityBoundingSet=~CAP_SYS_MODULE CAP_SYS_RAWIO CAP_SYS_TIME CAP_AUDIT_CONTROL
CAP_KILL CAP_MKNOD CAP_NET_BROADCAST CAP_NET_RAW
   CapabilityBoundingSet=~CAP_SYS_NICE CAP_MAC_ADMIN CAP_MAC_OVERRIDE CAP_BPF
CAP_SYS_BOOT CAP_LINUX_IMMUTABLE CAP_IPC_LOCK
   CapabilityBoundingSet=~CAP_BLOCK_SUSPEND CAP_LEASE CAP_SYS_PACCT CAP_SYS_TTY_CONFIG
CAP_WAKE_ALARM
```

- Die Anwendung darf sich nur an bestimmte TCP/UDP-Sockets binden (Ports für eingehenden Netzwerk-Verkehr öffnen) — Beispiel für einen DNS-Server:

```
   SocketBindDeny=any
   SocketBindAllow=53     # DNS
   SocketBindAllow=953    # RNDC
   SocketBindAllow=853    # DNS-over-TLS
```

2.16.4. Systemd-Unit-Firewall

- Systemd kann (mit der Hilfe von eBPF Programmen — siehe Kapitel eBPF) IPv4 und IPv6 Pakete auf der Prozess/Unit Ebene filtern:

```
   IPAddressDeny=any
   IPAddressAllow=172.16.1.0/24 192.168.178.64/27 192.0.2.22 100.64.2.53
```

- Die Firewall filtert sowohl eingehende- als auch ausgehende Pakete
  - Bei eingehenden Paketen wird die Quell-IP-Adresse mit der *Allow*-Liste verglichen
  - Bei ausgehenden Paketen wird die Ziel-IP-Adresse mit der *Allow*-Liste verglichen
  - Alle Adressen welche nicht explizit per *Deny*-Liste geblockt werden sind erlaubt

2.16.5. Aufgabe:

- Ergänze die System-Unit für `nginx.service` mittels `systemctl edit nginx.service` um die oben genannten Sicherheits-Erweiterungen. Prüfe die Sicherheitswertung mit `systemd-analyze security nginx.service` und stelle sicher, das der Dienst weiterhin startet

2.17. SYSTEMD JOURNAL

- Die "Logdaten" werden in einem Linux-System mit systemd in einer binären Datenbank mit umfassenden Suchwerkzeugen gespeichert
  - Diese Datenbank nennt sich `journal`
- Nachteile gegenüber traditionellen Log-Dateien
  - Kein KISS Design
  - Schlechte post-mortem Analyse
  - Nicht mehr kompatibel zu alten Logauswertungen (z.B. logwatch)
- Vorteile
  - Metainfos nicht mehr fälschbar (weil vom Daemon)
  - Journal-Einträge können (optional) durch Verkettung der Log-Einträge per kryptografischen Hashes gegen nachträgliche Änderungen der Datenbank geschützt werden
  - Wartungsfrei (kein logrotate)
  - Kann applikationsspezifische Werte aufnehmen
  - Umfangreiche Abfragemöglichkeiten

2.17.1. Dokumentation

- http://0pointer.de/blog/projects/journalctl.html [http://0pointer.de/blog/projects/journalctl.html]

2.17.2. Journal-Dateien

- `/var/log/journal/<machine-id>` ← persistent
- `/run/log/journal/<machine-id>` ← dynamisch

Die *machine-id* steht in `/etc/machine-id` und wird automatisch generiert oder mit `systemd-machine-id-setup` erzeugt. Das Verzeichnis `/var/log/journal` muss vorhanden sein; systemd-journald loggt andernfalls nur temporär (Datenbank in einer TMPFS-Ramdisk).

2.17.3. Benutzung von journalctl

- Alle Journalmeldungen anzeigen

```
journalctl
```

- An das Ende der Logdaten springen

```
journalctl -e
```

- Datei verfolgen (wie `tail -f`) inkl. allen Metadaten und Catalog-Meldungen

```
journalctl -f -a -x
```

- Log-Einträge eines bestimmten Dienstes anzeigen

```
journalctl _SYSTEMD_UNIT=ssh.service
journalctl -u ssh.service
journalctl /usr/sbin/sshd
```

- Kernel Meldungen (Alternative zu `dmesg`) anzeigen

```
journalctl -k
```

- Alle Felder der Log-Einträge aufschlüsseln (optional als JSON ausgeben)

```
journalctl -o verbose
journalctl -o json-pretty
```

(alle Felder, die mit '_' beginnen, sind interne Felder und werden intern vom journald gesetzt und nicht vom Client. Somit sind sie nicht leicht manipulierbar.)

- Alle Log-Meldungen seit dem letztem Boot

```
journalctl -b
```

- Alle Log-Meldungen in einem bestimmten Zeitraum

```
journalctl --since "2020-01-10" --until "2020-01-24 12:00"
```

- Log-Meldungen ab einem bestimmten Log-Level

```
journalctl -p 4
journalctl -p warning
```

- Log-Meldungen aus der Shell in das Journal schreiben

```
ls | systemd-cat
```

- Größe der Journal-Datenbank beschränken: in der Datei `/etc/systemd/journald.conf`:

```
SystemMaxUse=100M
SystemKeepFree=1G
```

2.17.4. Forward-Secure-Sealing (FSS)

- Forward Secure Sealing (FSS) ist eine Funktion im Journal, die dazu dient, Manipulationen von Protokolldateien zu erkennen. Da Angreifer oft versuchen, ihre Aktionen zu verbergen, indem sie Einträge in den Protokolldateien ändern oder löschen, bietet FSS den Administratoren einen Mechanismus, um solche unautorisierten Änderungen zu erkennen.
- Um FSS Benutzen zu können muss das Journal persistent sein

```
# mkdir /var/log/journal
# systemctl restart systemd-journald
# journalctl --flush
```

- FSS Schlüssel erstellen: FSS benötigt zwei kryptografische Schlüssel
  - Sealing-Schlüssel: Erzeugt Signaturen für die Journal-Einträge
  - Verifikation-Schlüssel: Wird benutzt um die Signaturen des Journal zu prüfen
  - Der *Sealing-Schlüssel* wird pro Interval neu aus dem vorherigen Schlüssel erzeugt. Der *Verifikation-Schlüssel* muss dabei nicht erneuert werden. Innerhalb des Interval-Zeitraums kann ein Angreifer den aktuellen *Sealing-Schlüssel* im Dateisystem finden und zur Manipulation des Journals benutzen. Erst nach dem Erzeugen eines neuen *Sealing-Schlüssels* ist der vorherige Schlüssel nicht mehr verfügbar und die Journal-Einträge geschützt. Das Sealing-Interval sollte daher möglich kurz gewählt werden.

```
# journalctl --setup-keys
Generating seed...
Generating key pair...
Generating sealing key...
Unable to set file attribute 0x1 on n/a, ignoring: Operation not supported
Unable to set file attribute 0x800000 on n/a, ignoring: Operation not supported
+
New keys have been generated for host selinux015/a7f85404d5a54e2392d6a92fa98be15a.
+
The secret sealing key has been written to the following local file.
This key file is automatically updated when the sealing key is advanced.
It should not be used on multiple hosts.
+
        /var/log/journal/a7f85404d5a54e2392d6a92fa98be15a/fss
+
The sealing key is automatically changed every 15min.
+
Please write down the following secret verification key. It should be stored
in a safe location and should not be saved locally on disk.
+
        436e17-ca6a3d-4310a2-345c98/1d48b1-35a4e900
```

- Der *Verifikation-Key* muss vom Admin sicher archiviert werden, um hiermit später das Journal prüfen zu können
- Optionen beim Erstellen der FSS-Schlüssel
  - `--interval=10s` Key-Roatation Intervall (Default: 15 Minuten). Innerhalb dieses Intervall kann ein Angreifer die Logs manipulieren. Kleinere Interval-Zeiträume sind sicherer, verbrauchen aber mehr CPU-Ressourcen da häufiger neue Schlüssel generiert werden

- `--force` Erzwingt die Neu-generierung eines FSS Schlüsselpaars

  - Journal-Integrität prüfen:

```
# journalctl --verify --verify-key=820ff0-19141a-983adc-229fdf/1dd3e0-35a4e900
PASS:
/var/log/journal/ab2fd7fc1dcf4914aedf7f14c2455bd5/system@0bfd1906769645fd8b567188f60176
21-00000000000046e7-00063ff3f5276215.journal
PASS:
/var/log/journal/ab2fd7fc1dcf4914aedf7f14c2455bd5/system@0bfd1906769645fd8b567188f60176
21-0000000000000001-00063fe08037565f.journal
PASS:
/var/log/journal/ab2fd7fc1dcf4914aedf7f14c2455bd5/system@0bfd1906769645fd8b567188f60176
21-00000000000090b5-00063fff6aa89596.journal
PASS: /var/log/journal/ab2fd7fc1dcf4914aedf7f14c2455bd5/system.journal
PASS: /var/log/journal/ab2fd7fc1dcf4914aedf7f14c2455bd5/user-1000.journal
```

- Journal FSS Dokumentation:
  - https://lwn.net/Articles/512895/ [https://lwn.net/Articles/512895/]
  - Practical Secure Logging: Seekable Sequential Key Generators (Giorgia Azzurra Marson and Bertram Poettering) https://eprint.iacr.org/2013/397 [https://eprint.iacr.org/2013/397]
  - Secure Logging in between Theory and Practice: Security Analysis of the Implementation of Forward Secure Log Sealing in Journald (Felix Dörre and Astrid Ottenhues) https://eprint.iacr.org/2023/867.pdf [https://eprint.iacr.org/2023/867.pdf]

2.17.5. Erweiterte Journal Einstellungen und Parameter

- die Journal-Einstellungen `systemd.journald.max_level_console`, `systemd.journald.max_level_store`, `systemd.journald.max_level_syslog`, `systemd.journald.max_level_kmsg`, `systemd.journald.max_level_wall` können über die Kernel-Kommandozeile übergeben werden und sind aktiv, wenn das Journal in der Init-Ramdisk gestartet wird (und die Journal-Konfigurarionsdatei noch nicht gelesen wurde).
- Maximale Anzahl von Journal-Datenbank-Dateien: in der Konfiguration `/etc/systemd/journal.conf` kann konfiguriert werden, wieviel archivierte Journal-Datenbank-Dateien der Journal-Daemon vorhalten soll. Der Standardwert ist 100. Dieser Wert kann über die Einstellungen `SystemMaxFiles` und `RuntimeMaxFiles` angepasst werden. Mit dem Befehl `journalctl --vacuum-files` können alle alten Journal-Datenbanken bis auf die im Befehl angegebene Anzahl gelöscht werden.
- Der Befehl `journalctl --sync` sort dafür das der Journal-Daemon alle noch im Speicher befindlichen Meldungen in die Datenbank im Dateisystem schreibt
- Wird beim Befehl `journalctl` als Selektor ein Gerätepfad angegeben, so gibt der Befehl auch Meldungen der Eltern-Geräte-Treiber aus (z.B. des SATA-Kontrollers, wenn eine SATA-Platte angegeben wird)
- Mit dem Parameter `journalctl --root` kann ein Journal aus einem alternativen Root-Verzeichnis (z.B. Container) gelesen werden. In den Verzeichnis muss kein Journal-Dienst gestartet sein (d.h. bei einem Container muss der Container nicht gestartet sein)
- Mit der Option `--grep=<suchbegriff>` kann die Ausgabe von `journalctl` auf einen beliebigen Suchbegriff gefiltert werden. Diese Filterung geschied im Journal-Daemon und ist

effizienter als eine nachträgliche Filterung mit `grep` auf der Kommandozeile

2.17.6. Journal-Speicherplatz

- Dateisystem-Platzverbrauch des Systemd-Journals abfragen

```
# journalctl --disk-usage
Archived and active journals take up 4.0G in the file system.
```

- Journal-Datenbank verkleinern

```
# journalctl --vacuum-size=500M
# journalctl --disk-usage
Archived and active journals take up 488.1M in the file system.
```

2.18. REMOTE LOGGING MIT JOURNALD

- Vorteile von Journald Logging im Vergleich zu Syslog
  - Mehr Struktur und Metadaten in den Meldungen
  - Datenbankabfrage nach Meldungen, einfache Korrelierung von Events über mehere Systeme
  - Transport über HTTPS/TLS, auch über Web-Proxies möglich
  - (Optional) Abgeschlossene (sealed = kryptografisch abgesichterte) Journal-Datenbanken
  - Pull oder Push Kommunikation möglich

2.18.1. Dienst systemd-journal-gatewayd

- Mit dem Dienst `systemd-journal-gatewayd` kann das Journal eines Rechners nach aussen über das Netzwerk exportiert werden. Die Daten werden über HTTP- oder HTTPS-Protokoll ausgeliefert. Für HTTPS muss ein x509-Zertifikat hinterlegt werden.
- Paket für Journal-Logging über Netzwerk installieren. Das Paket `systemd-journal-remote` beinhaltet den `systemd-journal-gatewayd` Dienst

```
# apt install systemd-journal-remote
```

- Dienst-Socket starten. Der Socket horcht auf Port 19531/TCP:

```
# systemctl enable --now systemd-journal-gatewayd.socket
```

- System-Reboot oder Rechte auf `/var/log/journal` für Gruppe `systemd-journal` setzen (z.B. via `systemd-tmpfiles --create`)
- Per Browser auf `http://<rechnername>:19531/` ist nun das Journal über ein Web-GUI abfragbar (Achtung: es gibt keine Benutzer-Authentisierung!)
- Journal-Einträge können auch über HTTP-Kommandozeilen Programme wie `curl` oder `wget` abgerufen werden

```
# curl 'http://<rechnername-oder-ip>:19531/entries?follow'
```

- Dabei können die Einträge auch auf Journal-Felder gefiltert werden

```
curl 'http://<rechnername-oder-ip>:19531/entries?follow&_COMM=sshd'
```

- `systemd-journal-gatewayd` ist die Gegenstelle zu einem zentralen Journal im **Pull** Modus

2.18.2. systemd-journal-remote.service

- Das Journal kann in zwei verschiedenen Modi über das Netzwerk transportiert werden: Pull-Modus (zentraler Server holt die Daten von den zu überwachenden Rechnern) oder Push-Modus (zu überwachende Rechner senden aktiv die Daten zum zentralen Journal-Server)

Pull Modus / Aktiver Modus

- Auf dem zentralen Journal-Log-Server das Paket für Journal-Logging über Netzwerk installieren

```
# apt install systemd-journal-remote
```

- Ein Zielverzeichnis für die neuen Logs erstellen

```
# mkdir -p /var/log/journal/remote
```

- Manueller Test, ob die Daten vom entfernten System gelesen werden können (Test nach einigen Sekunden mit CTRL+C abbrechen)

```
# /usr/lib/systemd/systemd-journal-remote --url http://<rechnername-oder-
ip>:19531/entries?follow
```

- Prüfen das eine neue Journal-Datenbank angelegt wurde

```
# ls -l /var/log/journal/remote/
```

- Remote Logs anschauen

```
# journalctl -D /var/log/journal/remote -lf
```

- Lokales Journal zusammen (Parameter `-m` Merge) mit den entfernten Journals durchsuchen (hier nach Meldungen der Unit `sshd`)

```
# journalctl -lmu sshd
```

Push Modus / Passiver Modus

- Empfänger/Log-Server
- Der Push-Modus erwartet verschlüsselte Kommunikation per **https**. Dafür werden im Internet gültige x509 Zertifikate oder Zertifikate aus einer eigenen CA benötigt. Um die Funktionalität zu zeigen, arbeiten wir hier mit unverschlüsseltem HTTP. Diese Konfiguration ist für Produktiv-Umgebungen **nicht** zu empfehlen (bitte Zertifikate besorgen und HTTPS benutzen!)
- Die Unit-Datei des Journal-Empfänger-Dienstes anzeigen

```
less /usr/lib/systemd/system/systemd-journal-remote.service
```

- Ein Konfigurations-Drop-In für die Systemd-Unit des Dienstes erstellen (für die

Konfigurationsänderung HTTPS zu HTTP)

```
# systemctl edit systemd-journal-remote.service
```

- Inhalt der Drop-In Datei

```
[Service]
ExecStart=
ExecStart=/usr/lib/systemd/systemd-journal-remote --listen-http=-3
--output=/var/log/journal/remote/
```

- Die Konfigurationsdatei des Dienstes in `/etc/systemd/journal-remote.conf`. Hier werden bei der Benutzung von HTTPS die Zertifikatsdateien eingetragen.

```
[Remote]
# Seal=false
# SplitMode=host
# ServerKeyFile=/etc/ssl/private/journal-remote.pem
# ServerCertificateFile=/etc/ssl/certs/journal-remote.pem
# TrustedCertificateFile=/etc/ssl/ca/trusted.pem
```

- Die neuen Journal-Dateien werden unter `/var/log/journal/remote` abgelegt, dieses Verzeichnis muss für den Benutzer `systemd-journal-remote` beschreibbar sein

```
# chown systemd-journal-remote /var/log/journal/remote
```

- Den Socket-Listener für den Remote-Journal Dienst starten

```
systemctl enable --now systemd-journal-remote.socket
```

- Der Dienst sollte nun auf dem Port 19532 horchen

```
# dnf install -y lsof
# lsof -Poni :19532
COMMAND    PID                   USER   FD   TYPE DEVICE OFFSET NODE NAME
systemd      1                   root   27u  IPv6  82082    0t0  TCP *:19532 (LISTEN)
systemd-j 2562 systemd-journal-remote    3u  IPv6  82082    0t0  TCP *:19532 (LISTEN)
```

Log-Sender (Log-Client)

- Konfigurationsdatei `/etc/systemd/journal-upload.conf`

```
[Upload]
URL=http://<rechnername-oder-ip>
# ServerKeyFile=/etc/ssl/private/journal-upload.pem
# ServerCertificateFile=/etc/ssl/certs/journal-upload.pem
# TrustedCertificateFile=/etc/ssl/ca/trusted.pem
```

- Manueller Test eines Uploads. Der Parameter `--save-state` speichert die Information, bis zu welchem Journal-Eintrag die Daten schon gesendet wurden

```
# /usr/lib/systemd/systemd-journal-upload --save-state
```

- Die State-Datei wieder löschen (diese gehört dem benutzer Root und kann vom System-journal-

upload Dienst nicht gelesen und geschrieben werden)

```
# rm /var/lib/systemd/journal-upload/state
```

- War der manuelle Test erfolgreich, den Journal-Upload Dienst starten

```
# systemctl enable --now systemd-journal-upload
```

2.19. SICHERHEIT VON DATEISYSTEM-MOUNTS

- Die Sicherheit eines Linux-Systems kann durch optionale Mount-Optionen erhöht werden
  - Read-Only Mounts: `mount -r …` oder `mount -o ro` – Sind wichtige Teile des Betriebssystems "read-only" gemounted (z.B. `/usr` oder `/etc`) so können Dateien dort weniger einfach manipuliert werden. Dies gilt insb. für *Bind-Mounts* aus dem Linux "Host"-System in einen Linux-Namespace/Container
  - SELinux File-Label/Context: `mount -o context=context …`, `fscontext=context`, `defcontext=context` und `rootcontext=context`: SELinux benötigt Context-Daten für jede Datei im Dateisystem. Diese Daten werden bei XFS-, ext4- und btrfs-Dateisystemen in Erweiterten Attributen im Dateisystem gespeichert. Einige Dateisysteme wie NFSv2/v3, exFAT, CIFS haben jedoch keine Möglichkeit, diese erweiterten Attribute zu speichern. Mit diesen Mount-Optionen kann ein SELinux-Security-Context für alle Dateien im Dateisystem vorgegeben werden.
  - Keine ausführbaren Dateien: `mount -o noexec …`: ausführbare Dateien können von diesem Dateisystem aus nicht ausgeführt werden. Sollte für Daten-Dateisysteme verwendet werden, speziell wenn Benutzer Daten und damit Programme von aussen in dieses Dateisystem speichern können.
  - Keine Gerätedateien: `mount -o nodev …`: Block- oder Zeichenbasierte (Charakter) Gerätedateien (welche normalerweise unterhalb von `/dev` zu finden sind) haben in diesem Dateisystem keine besondere Funktion und erlauben keinen Zugriff auf die Hardware oder Kernel-Funktionen.
  - Keine SUID/SGID-Dateien: `mount -o nosuid …`: Das *SUID*- oder das *SGID*-Bit von Dateien wird auf diesem Dateisystem ignoriert, d.h. werden Programme von diesem Dateisystem ausgeführt so können diese keine weiteren Rechte durch die *SUID*- oder *SGID*-Bits erlangen
- *BIND*-Mounts: In einem Linux-System können bestehende Verzeichnisse und auch einzelne Dateien mittels eines *BIND-Mounts* an beliebige Stellen im Verzeichnisbaum eingehangen werden.
  - Dabeikönnen dabei sicherheits-relevante Mount-Optionen angepasst werden
  - Der Parameter `--bind` hängt ein bestehendes Verzeichnis oder Datei an einen anderen Mount-Punkt. Weitere Mounts im Quell-Dateisystem werden jedoch nicht durchgereicht
  - Der Parameter `--rbind` (rekursiver BIND-Mount) hängt ein Verzeichnis und alle darunter eingehangenen Dateisystem an den neuen Mount-Punkt.
  - Beispiele:
    - Read-Only Mounts aus dem Linux-Host in einen Linux-Namespace/Container
    - Dateien und Verzeichnisse aus Netzwerklaufwerken in den Haupt-Dateisystem-Baum mounten

  - Programm-Verzeichnisse mit `noexec` mounten, um Benutzer die Option zur Analyse der Dateien zu geben, ohne Gefahr das diese ausgeführt werden (z.B. bei Malware-Analyse)

- BIND-Mount einer Datei: das Ziel eines BIND-Mounts einer Datei muss eine existierende Datei sein. In diesem Beispiel wird das Programm `sleep` über das Programm `top` gemounted:

```
$ sudo mount --bind /usr/bin/sleep /usr/bin/top
$ top 10
```

2.19.1. Aufgabe: BIND-Mount des checksec Verzeichnisses

- Wir arbeiten als Benutzer `user`, nicht als `root`
- Verezichnis für lokale Programmdateien anlegen

```
$ cd /home/user
$ mkdir -p /home/user/.local/bin
```

- Umgebungsvariable des Suchpfades für Programmdateien anpassen

```
$ export PATH=/home/user/.local/bin:$PATH
```

- Das `checksec.sh` Verzeichnis über das angelegte Verzeichnis mounten

```
$ sudo mount --bind /home/user/checksec.sh /home/user/.local/bin
```

- Prüfen das `checksec.bash` nun direkt gefunden wird

```
$ which checksec.bash
/home/user/.local/bin/checksec.bash
```

- Checksec aufrufen

```
$ checksec.bash --kernel
```

- Bind-Mount mit Option `noexec` neu anhängen

```
$ sudo mount -o noexec --bind -o remount /home/user/checksec.sh/
/home/user/.local/bin/
```

- `checksec.bash` ist nun nicht mehr ausführbar

```
$ checksec.bash
bash: /home/user/.local/bin/checksec.bash: Permission denied
```

2.20. LINUX CGROUPS (CONTROLL-GROUPS)

- CGroups Informationen anzeigen

```
cat /proc/cgroups
ps xawf -eo pid,user,cgroup,args
systemd-cgls
systemd-cgtop
```

2.20.1. CGroups manuell

- Stress erzeugen

```
apt install stress-ng
stress-ng --cpu $(($(nproc)*2)) &
pgrep -alf stress-ng
```

- CGroup manuell anlegen und kontrollieren

```
cd /sys/fs/cgroup
mkdir stressgroup
cd stressgroup
cat cgroup.controllers
cat cgroup.type
cat cgroup.subtree_control
```

- Neue Untergruppen anlegen und kontrollieren

```
mkdir stress1 stress2
ls -l stress1
cat stress1/cgroup.type
cat stress1/cgroup.controllers
```

- Controller vererben

```
echo +cpu > cgroup.subtree_control
ls -l stress1
```

- Alle Prozesse in die Gruppe stress1 legen.

```
cd stress1
pgrep stress-ng | while read pid; do echo $pid > cgroup.procs; done
```

- Schrittweise Last auf die Gruppe stress2 verteilen. Top beobachten. Weiter bis ca. Gleichstand erreicht ist.

```
cd ../stress2
echo <pid> > cgroup.procs
```

- Kontrolle mit weight. Top beobachten

```
echo 10 > cpu.weight
```

- In der anderen Gruppe mit harten Limits (CPU=100%) begrenzen

```
cd ../stress1
cat cpu.max
echo 100000 100000 > cpu.max
```

- Alle Prozesse beenden und Controll-Groups entfernen

```
pkill stress-ng
rmdir stress1 stress2
cd ..
```

```
rmdir stressgroup
```

2.20.2. systemd

- Controll-Group-Daten können bei durch Systemd gestartete Prozesse im laufenden Betrieb angepasst werden

```
systemctl set-property --runtime cups.service CPUQuota=10%
systemctl cat cups.service
```

2.20.3. Systemresourcen beschränken mit systemd-run

```
systemd-run stress-ng -c 3
systemd-run stress-ng -c 3
systemctl show run-r<UUID>.service
systemctl set-property run-r<UUID>.service CPUWeight=33
systemctl set-property run-r<UUID>.service CPUQuota=100%
systemd-run --on-active=20 -p CPUQuota=50% stress-ng --cpu 4
```

2.20.4. systemctl accounting anschalten

```
    $EDITOR /etc/systemd/system.conf.d/accounting.conf
    -----
    DefaultCPUAccounting=yes
    DefaultIOAccounting=yes
    DefaultBlockIOAccounting=yes
    DefaultMemoryAccounting=yes
    DefaultTasksAccounting=yes
```

- BlockIO... gilt als deprecated. Deshalb fehlt die Directive.

```
    systemd-analyze cat-config /etc/systemd/system.conf   # pruefen
```

2.21. DATEISYSTEMBERECHTIGUNGEN

2.21.1. Normale Unix-Rechte

die klassischen Unix-Dateisystemberechtigungen (RWX)

```
drwxr-xr-x 7 root root 4096 Sep 28 18:36 acme.sh
```

- Besitzer: RWX (R=Read, W=Write, X=Execute/Enter)
- Gruppe: RWX (R=Read, W=Write, X=Execute/Enter)
- Rest-der-Welt: RWX (R=Read, W=Write, X=Execute/Enter)

Das sgid-bit

```
     chmod g+s <verzeichnis>
```

Neue Dateien bekommen die Gruppe des Verzeichnisses anstatt der (primäre) Gruppe des Benutzers. Beim Anlegen von Unterverzeichnissen bekommen diese auch ein sgid bit gesetzt

Unerwartet: Schreibrechte auf Verzeichnis vs. Schreibrechte auf Datei

```
useradd nutzer1
useradd nutzer2
groupadd projekt
usermod -a -G projekt nutzer1
usermod -a -G projekt nutzer2
mkdir /home/projekt
chown :projekt /home/projekt
chmod g+w /home/projekt
ls -ld /home/projekt/
su - nutzer1
cd /home/projekt/
cat > unveraenderbarer.txt
Dies ist ein unveraenderbarer text
CTRL+D
chmod u=rw,g=r,o=r unveraenderbarer.txt
chgrp projekt unveraenderbarer.txt
ls -l unveraenderbarer.txt
-rw-r--r--. 1 cas projekt 35 Nov 23 21:57 unveraenderbarer.txt
logout
su - nutzer2
cd /home/projekt
ls -l unveraenderbarer.txt
vi unveraenderbarer.txt
```

Frage:

- kann `nutzer2` mit diesen Berechtigungen die Datei verändern?

Antwort:

- ja, indirekt. Der Benutzer hat Schreibrechte auf dem Verzeichnis. Daher kann er neue Dateien anlegen und bestehende Löschen. Er kann auch Dateien löschen, auf denen er keine Schreibrechte besitzt! Der `vi` Editor legt bei Bearbeiten einer Datei eine lokale Kopie an, diese Kopie wird editiert und beim Speichern wird die Ausgangsdatei gelöscht und durch die temporäre Datei ersetzt.

Sticky-Bit auf Verzeichnis – Nur der Besitzer der Datei darf die Datei löschen

```
    chmod o+t <verzeichnis>
```

2.21.2. Erweiterte Attribute

- Nicht alle Dateisysteme unterstützen alle erweiterten Attribute (EA). Unter Linux unterstützen die ext2/ext3/ext4/btrfs-Dateisysteme und das XFS-Dateisystem einige der EAs. Die Man-Page zum Dateisystemformat gibt Auskunft über die EA Unterstützung (z.B. `man ext4`).

- Anzeigen der erweiterten Attribute: `lsattr`
- Ändern der erweiterten Attribute: `chattr`
- Sicherheit von erweiterten Attribute (EA) unter Linux im Vergleich zu BSD: unter BSD können die Sicherheitsrelevanten EA (append, immutable) vom Benutzer **root** nur im Single–User Module (ohne Netzwerk) gelöscht werden. Unter Linux kann **root** die EA zu jedem Zeitpunkt entfernen!

Erweiterte Attribute vom Benutzer verwaltbar

- d□—□Datei nicht bei "dump" (Backup) mitsichern
- s□—□Datei wird beim Löschen mit Nullen überschrieben
- A□—□bei der Datei wird *atime* (letztes Zugriffsdatum) nicht aktualisiert

Erweiterte Attribute welche nur vom Benutzer root verwaltet werden

- a□—□append – Datei kann nur erweitert werden
- i□—□immutable – Datei ist unveränderbar
- C□—□no copy–on–write – für COW–Filesysteme (btrfs)

Beispiele

- Werkzeuge zum Setzen von Posix ACL installieren

```
apt install acl
```

```
# Erweiterte Attribute auslesen
lsattr /pfad/zur/datei
# Erweitertes Attribut "d" setzen
chattr +d /pfad/zur/datei
# Erweitertes Attribut "d" löschen
chattr -d /pfad/zur/datei
```

2.21.3. Posix ACLs

ACLs setzen (modify)

```
    setfacl -m u:<nutzer>:rwx file/dir          # ACL setzen
    setfacl -x u:<nutzer>                       # ACL löschen
    setfacl -d -m "u::rwx,g::rwx,o::-" <dir>    # Default ACL auf Verzeichnisse setzen
    setfacl -m "d:u::rwx,d::g:rwx,d:o::-" <dir> # alternative Syntax. Generic default
mask

    setfacl -d -m u:<uidName>:rwx <file/dir>    # Benamte Default ACL macht sie
"vererbbar"
```

ACLs auslesen

```
getfacl <datei/verzeichnis>
```

ACL mask

- `chmod gruppe` ändert nur noch die Maske
- die Maske filtert die ACLs
- Berechtigungs-Änderungen können nur mit `setfacl` durchgeführt werden
- setfacl -k □—□default ACLs löschen
- setfacl -b □—□ACLs löschen

2.22. PAM-PLUGGABLE AUTHENTICATION MODULES

- Dokumentation des Linux-PAM-Systems
  - https://github.com/linux-pam/linux-pam/ [https://github.com/linux-pam/linux-pam/]
  - (nicht mehr vorhanden) http://www.linux-pam.org/Linux-PAM-html/Linux-PAM_SAG.html [http://www.linux-pam.org/Linux-PAM-html/Linux-PAM_SAG.html]
  - https://fossies.org/linux/Linux-PAM-docs/doc/sag/Linux-PAM_SAG.pdf [https://fossies.org/linux/Linux-PAM-docs/doc/sag/Linux-PAM_SAG.pdf]
  - (als Archiv, enthält auch das alte HTML SAG) https://fossies.org/linux/misc/Linux-PAM-1.6.1-docs.tar.xz/ [https://fossies.org/linux/misc/Linux-PAM-1.6.1-docs.tar.xz/]
- Pfad für PAM-Module (bis Debian 11) `/lib/x86_64-linux-gnu/security/`
- Pfad für PAM-Module (ab Debian 12) `/usr/lib/x86_64-linux-gnu/security/`
- Beispiel der PAM Konfigurationsdatei `/etc/pam.d/common-auth`

```
cat /etc/pam.d/common-auth | grep -v "#"

auth    [success=1 default=ignore]      pam_unix.so nullok_secure
auth    requisite                       pam_deny.so
auth    required                        pam_permit.so
auth    optional                        pam_cap.so
```

- PAM Dienst-Typen
  - `account`: Prüfung Berechtigung
  - `auth`: Authentifizierung
  - `password`: Passwort-Änderung
  - `session`: Sitzungsverwaltung
- PAM Control
  - `requisite` = muss erfolgreich sein, sonst Kette beenden (notwendige Vorbedingung)
  - `required` = muss am Ende erfolgreich sein (notwendige Bedingung)

    - `sufficient` = bei Erfolg wird die Kette beendet (hinreichende Bedingung)
    - `optional` = Returncode wird nicht verwendet
- Linux–PAM erweiterte Controls
    - Syntax: `[return-value=action …]`
    - Return–Values
        - vgl. z.B. `/usr/include//security/_pam_types.h`
    - Actions:
        - `OK` = Zugriff erlauben
        - `ignore` = Ignorieren
        - `bad` = Zugriff verweigern
        - `die` = Zugriff verweigern und Kette abschliessen
        - `done` = Zugriff erlauben und Kette abschliessen
        - `reset` = PAM Variablen zurücksetzen/löschen und weitermachen
        - `<n>` = die folgenden <n> PAM–Regeln überspringen

2.22.1. Aufgabe: `pam_warn.so` für sshd installieren

- Installiere das Modul `pam_warn.so` fuer die Facility `auth` (zusätzlich `session`) in PAM–Dienst `sshd`
- Melde dich per `ssh` oder über das Web–Portal als Benutzer `user` an
- Prüfe die Syslog/Journal Ausgabe

2.22.2. Lösung

- In Datei `/etc/pam.d/sshd` (ans Ende hinter allen @include)

```
auth       optional   pam_warn.so
session    optional   pam_warn.so
```

- Die neuen Log–Einträgen mit `journalctl -f --grep pam_warn` oder `journalctl -f --facility=authpriv` prüfen
- Als `user` einloggen

2.22.3. Aufgabe: ein einfaches PAM–Modul aktivieren

- Es gibt ein PAM–Modul, welches allen normalen Benutzern die Anmeldung am System verweigert. Nur der `root` Benutzer darf sich dann anmelden.
- Dieses PAM–Modul ist schon für den Dienst **sshd** konfiguriert, wirkt sich aber nicht aus (Datei `/etc/pam.d/sshd`)
- Lese die Man–Pages der PAM–Module für den Dienst **sshd** der **auth**–Funktionen, finde heraus, welches Modul das Login aller Nicht–Root–Benutzer verweigern kann, und wann es sich auswirkt.

- Aktiviere diese Funktion und teste diese aus. Versuche Dich mit dem normalen Benutzer per SSH über das Loopback–Interface anzumelden:

```
ssh user@localhost
```

- Wie kann einem normalen Benutzer der Grund für den Fehlschlag des Anmeldeversuches mitgeteilt werden?

2.22.4. Lösung:

- Modul `pam_nologin.so`, aktiviert durch die Datei `/etc/nologin`:

```
echo 'Heute kein Login! Wartungsarbeiten bis Dienstag!' > /etc/nologin
```

2.22.5. Aufgabe: 2–Faktor Authentisierung – OATH Open Authentication Event Token (HOTP)

- RFC 4226 HOTP: An HMAC–Based One–Time Password Algorithm (https://tools.ietf.org/html/rfc4226 [https://tools.ietf.org/html/rfc4226])
- Alternative: RFC 6238 "TOTP: Time–Based One–Time Password Algorithm"
- Token–Software als App für viele Mobiltelefone verfügbar
- Pakete installieren

```
apt install libpam-oath oathtool
```

- `pam_oath` in die PAM Konfiguration aufnehmen (hier für den su Befehl). `window=5` gibt ein Fenster von 5 Passwörtern aus der Liste an, welche akzeptiert werden.

```
$EDITOR /etc/pam.d/su
-----
#%PAM-1.0
auth            sufficient      pam_rootok.so
auth            requisite       pam_oath.so usersfile=/etc/oath/users.oath window=5
-----
```

- OATH Benutzerdatei anlegen

```
mkdir /etc/oath
$EDITOR /etc/oath/users.oath
-----
HOTP user - <hex-secret>
HOTP nutzerYY - 0102030405
```

- Beispiel: Passwort in HEX–Zahl umrechnen:

```
echo "secret" | od -x
0000000 6573 7263 0a74
0000006
```

- Benutzerrechte setzen

```
chmod 000 /etc/oath/users.oath
chown root: /etc/oath/users.oath
```

- Eine Reihe (5 Stück) von Passwörter zum Test erstellen

```
oathtool -w 5 <hex-secret>
```

- Anmeldung ausprobieren als "user", eines der Passwörter aus der Liste probieren

```
su - user
```

- Wer ein Smart-Phone hat, mal im App-Store nach "OATH" suchen, eines OATH-Token Programm installieren und konfigurieren

2.22.6. PAM Duress

- *PAM Duress* (https://github.com/nuvious/pam-duress [https://github.com/nuvious/pam-duress]) ist ein interessantes PM-Modul welches es dem Benutzer erlaubt, alternative Passwörter im PAM zu hinterlegen. Diese Passwörter sind jeweils mit einen Shell-Skript verbunden. Wird eines der "Duress" Passwörter statt dem "normalen" Benutzerpasswort eingegeben, so wird das dazugehörige Script ausgeführt.
- Beschreibung der Einsatzszenatrien von der Projekt-Webseite

> Diese Funktion könnte genutzt werden, um jemandem, der unter [Zwang] zur Eingabe eines Kennworts gezwungen wird, die Möglichkeit zu geben, ein Kennwort einzugeben, das den Zugang gewährt, aber im Hintergrund Skripte ausführt, um sensible Daten zu bereinigen, Verbindungen zu anderen Netzwerken zu schließen, um *lateral movement* einzuschränken, und/oder eine Benachrichtigung oder einen Alarm zu senden (möglicherweise mit detaillierten Informationen wie Standort, sichtbaren WLAN-Hotspots, einem Bild von der Kamera, einem Link zu einem Stream vom Mikrofon usw.). Es kann sogar einen Prozess starten, um das Modul pam_duress zu entfernen, damit der Bedrohungsakteur nicht sehen kann, ob das PAM-Duress verfügbar war.

- Weitere Einsatzgeniete:
  - Wegwerfbare Gast-Zugänge einrichten
  - Automatisches mit-protokollieren einer Sitzung per "sudo" beim Login einschalten
  - MacOS "Find-my-Mac" nachbauen (gestohlene Rechner orten)

2.22.7. Weitere Links zu PAM

- Understanding the effects of PAM module results ('controls' in PAM jargon) https://utcc.utoronto.ca/\~cks/space/blog/sysadmin/PAMModuleResultsEffects]
- A Linux PAM setup and the problem of stopping authentication[https://utcc.utoronto.ca/~cks/space/blog/linux/PAMStackingAndStopping]
- TOTP authenticator implement as a terminal tool https://github.com/MinaOTP/MinaOTP-Shell [https://github.com/MinaOTP/MinaOTP-Shell]
- Minimal TOTP generator in 20 lines of Python https://github.com/susam/mintotp [https://github.com/susam/mintotp]
- Using a Yubikey as a touchless, magic unlock key for Linux https://kliu.io/post/yubico-magic-

unlock/ [https://kliu.io/post/yubico-magic-unlock/]

- Windows Hello™ style facial authentication for Linux https://github.com/boltgolt/howdy [https://github.com/boltgolt/howdy]

2.23. AUDIT SUBSYSTEM

- Das Audit-Subsystem überwacht Dateien und Systemaufrufe (Systemcalls) und schreibt Log-Informationen in das Audit-Log
- Audit Subsystem Homepage https://people.redhat.com/sgrubb/audit/ [https://people.redhat.com/sgrubb/audit/]

2.23.1. Installation

- Audit-Subsystem Programme installieren

```
apt install auditd audispd-plugins
```

2.23.2. Konfiguration

- Audit-Daemon Konfiguration anpassen (Grössen-Werte in MB)

```
$EDITOR /etc/audit/auditd.conf
----
space_left=4000
admin_space_left=2000
num_logs=10
max_log_file=<max-groesse-der-log-datei>
max_log_file_action=rotate
```

- Audit-Dämon anschalten (so dass er bei einem Restart neu gestartet wird) und prüfen das der Prozess läuft

```
systemctl enable --now auditd
systemctl status auditd
```

- Eine Audit-Policy erstellen

```
$EDITOR /etc/audit/rules.d/audit.rules
```

- Eine Beispiel-Policy:

```
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 0

## Set failure mode to syslog
```

```
-f 1

# audit_time_rules - Record attempts to alter time through adjtime
-a always,exit -F arch=b64 -S adjtimex -k audit_time_rules

# audit_time_rules - Record attempts to alter time through settimeofday
-a always,exit -F arch=b64 -S settimeofday -k audit_time_rules

# audit_time_rules - Record Attempts to Alter Time Through stime
-a always,exit -F arch=b64 -S adjtimex -S settimeofday -S clock_settime -k
audit_time_rules

# audit_time_rules - Record Attempts to Alter Time Through clock_settime
-a always,exit -F arch=b64 -S clock_settime -k audit_time_rules

# Record Attempts to Alter the localtime File
-w /etc/localtime -p wa -k audit_time_rules

# Record Events that Modify User/Group Information
# audit_account_changes
-w /etc/group -p wa -k audit_account_changes
-w /etc/passwd -p wa -k audit_account_changes
-w /etc/gshadow -p wa -k audit_account_changes
-w /etc/shadow -p wa -k audit_account_changes
-w /etc/security/opasswd -p wa -k audit_account_changes

# Record Events that Modify the System's Network Environment
# audit_network_modifications
-a always,exit -F arch=b64 -S sethostname -S setdomainname -k
audit_network_modifications
-w /etc/issue -p wa -k audit_network_modifications
-w /etc/issue.net -p wa -k audit_network_modifications
-w /etc/hosts -p wa -k audit_network_modifications

#Record Events that Modify the System's Mandatory Access Controls
-w /etc/selinux/ -p wa -k MAC-policy

#Record Events that Modify the System's Discretionary Access Controls - chmod
-a always,exit -F arch=b32 -S chmod -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S chmod  -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - chown
-a always,exit -F arch=b32 -S chown -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S chown -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - fchmod
-a always,exit -F arch=b32 -S fchmod -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S fchmod -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - fchmodat
-a always,exit -F arch=b32 -S fchmodat -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S fchmodat -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - fchown
```

```
-a always,exit -F arch=b32 -S fchown -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S fchown -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - fchownat
-a always,exit -F arch=b32 -S fchownat -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S fchownat -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - fremovexattr
-a always,exit -F arch=b32 -S fremovexattr -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S fremovexattr -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - fsetxattr
-a always,exit -F arch=b32 -S fsetxattr -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S fsetxattr -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - lchown
-a always,exit -F arch=b32 -S lchown -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S lchown -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - lremovexattr
-a always,exit -F arch=b32 -S lremovexattr -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S lremovexattr -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - lsetxattr
-a always,exit -F arch=b32 -S lsetxattr -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S lsetxattr -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - removexattr
-a always,exit -F arch=b32 -S removexattr -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S removexattr -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Events that Modify the System's Discretionary Access Controls - setxattr
-a always,exit -F arch=b32 -S setxattr -F auid>=500 -F auid!=4294967295 -k perm_mod
-a always,exit -F arch=b64 -S setxattr -F auid>=500 -F auid!=4294967295 -k perm_mod

#Record Attempts to Alter Logon and Logout Events
-w /var/log/faillog -p wa -k logins
-w /var/log/lastlog -p wa -k logins

#Record Attempts to Alter Process and Session Initiation Information
-w /var/run/utmp -p wa -k session
-w /var/log/btmp -p wa -k session
-w /var/log/wtmp -p wa -k session

#Ensure auditd Collects Unauthorized Access Attempts to Files (unsuccessful)
-a always,exit -F arch=b32 -S creat -S open -S openat -S open_by_handle_at -S truncate
-S ftruncate -F exit=-EACCES -F auid>=500 -F auid!=4294967295 -k access
-a always,exit -F arch=b32 -S creat -S open -S openat -S open_by_handle_at -S truncate
-S ftruncate -F exit=-EPERM -F auid>=500 -F auid!=4294967295 -k access
-a always,exit -F arch=b64 -S creat -S open -S openat -S open_by_handle_at -S truncate
-S ftruncate -F exit=-EACCES -F auid>=500 -F auid!=4294967295 -k access
-a always,exit -F arch=b64 -S creat -S open -S openat -S open_by_handle_at -S truncate
-S ftruncate -F exit=-EPERM -F auid>=500 -F auid!=4294967295 -k access
```

```
#Ensure auditd Collects Information on the Use of Privileged Commands
#
#  Find setuid / setgid programs then modify and uncomment the line below.
#
##  sudo find / -xdev -type f -perm -4000 -o -perm -2000 2>/dev/null
#
# -a always,exit -F path=SETUID_PROG_PATH -F perm=x -F auid>=500 -F auid!=4294967295 -k
privileged

#Ensure auditd Collects Information on Exporting to Media (successful)
-a always,exit -F arch=b64 -S mount -F auid>=500 -F auid!=4294967295 -k export

#Ensure auditd Collects File Deletion Events by User
-a always,exit -F arch=b64 -S rmdir -S unlink -S unlinkat -S rename -S renameat -F
auid>=500 -F auid!=4294967295 -k delete

#Ensure auditd Collects System Administrator Actions
-w /etc/sudoers -p wa -k actions

#Ensure auditd Collects Information on Kernel Module Loading and Unloading
-w /sbin/insmod -p x -k modules
-w /sbin/rmmod -p x -k modules
-w /sbin/modprobe -p x -k modules
-a always,exit -F arch=b64 -S init_module -S delete_module -k modules

#Make the auditd Configuration Immutable
-e 2
```

- Die Policy kompilieren und prüfen

```
augenrules --check
```

- die neue Policydatei in den Kernel laden

```
augenrules --load
```

- aktive Regeln auflisten

```
auditctl -l
```

- Status des Audit-Subsystems anzeigen

```
# auditctl -s
enabled 2
failure 1
pid 12901
rate_limit 0
backlog_limit 8192
lost 0
backlog 0
backlog_wait_time 0
loginuid_immutable 0 unlocked
```

2.23.3. Beispiele für Audit-Abfragen

- Alle Audit-Einträge zum Thema "sudo" zeigen

```
ausearch -i -x sudo
```

- Report über fehlgeschlagende Anmeldeversuche

```
ausearch -m USER_AUTH,USER_ACCT --success no
```

- Alle Audit-Meldungen für Benutzer UID 1000

```
ausearch -ua 1000 -i
```

- Fehlgeschlagende Syscalls seit gestern

```
ausearch --start yesterday --end now -m SYSCALL -sv no -i
```

Aufgabe:

- Das Audit-Subsystem um neue Regel(n) zu Systemd erweitern:
  - Alle Systemd-Konfigurationsdateien mit der Endung `*.conf`, welche direkt unter `/etc/systemd` (nicht in den Unterverzeichnissen) gespeichert sind, sollen auf Schreibzugriffe überwacht werden
  - Die Audit-Events sollen mit dem Key `systemd` markiert werden
  - Die neue Richtlinie kompilieren und aktivieren (ggf. Reboot notwendig)
  - Die neue Richtlinie testen, in dem manuell an einer der überwachten Dateien eine Änderung vorgenommen wird
  - Die Audit-Meldungen per `ausearch` anzeigen und analysieren (suchen nach Events mit dem Key `systemd`)

Beispiel-Lösung

- Erweiterung der Audit-Regeln

```
[...]
# Systemd Konfiguration
-w /etc/systemd/journald.conf -p wa -k systemd
-w /etc/systemd/logind.conf -p wa -k systemd
-w /etc/systemd/networkd.conf -p wa -k systemd
-w /etc/systemd/resolved.conf -p wa -k systemd
-w /etc/systemd/sleep.conf -p wa -k systemd
-w /etc/systemd/system.conf -p wa -k systemd
-w /etc/systemd/timesyncd.conf -p wa -k systemd
-w /etc/systemd/user.conf -p wa -k systemd
[...]
```

- Suche nach Audit-Events mit dem Schlüssel `systemd`

```
ausearch -i -k systemd
```

2.23.4. Audit-Hilfsprogramme

- Zusammenfassung des Audit-Log

```
aureport
```

- Grafische Auswertung von Audit-Logs

```
$ sudo apt install graphviz gnuplot git
$ git clone https://github.com/cstrotm/audit-visualize
$ cd audit-visualize
$ chmod +x ./mkgraph
$ chmod +x ./mkbar
```

- Grafische Reports erstellen

```
$ sudo aureport -s -i --summary  | bash ./mkbar syscall
$ sudo aureport -f -i --summary --failed | bash ./mkbar failed-access
$ sudo aureport -e -i --summary | egrep -vi '(syscall|change)'
$ sudo aureport -e -i --summary | egrep -vi '(syscall|change)' | bash ./mkbar events2
```

- Syscall Benutzung von Programmen

```
$ sudo aureport -s -i | awk '/^[0-9]/ { printf "%s %s\n", $6, $4 }' | sort | uniq |
bash ./mkgraph
Gzipping graph...
Graph was written to gr.png
```

- Welcher Benutzer führt welche Programme aus?

```
sudo aureport -u -i | awk '/^[0-9]/ { printf "%s %s\n", $4, $7 }' | sort | uniq | bash
./mkgraph
```

- Wer greift auf welche Dateien zu?

```
sudo aureport -f -i | awk '/^[0-9]/ { printf "%s %s\n", $8, $4 }' | sort | uniq | bash
./mkgraph
```

2.23.5. CIS Benchmarks mit Vorlagen für Audit-Regeln

- Webseite: https://www.cisecurity.org/cis-benchmarks/ [https://www.cisecurity.org/cis-benchmarks/]

2.24. DATEISYSTEM-VERSCHLÜSSELUNG

- In diesem Kapitel berachen wir populäre und expemplarische Vertreter von Dateisystem-Verschlüsselungs-Lösungen unter Linux
  - **ext4**: Verschlüsselung im Dateisysten, unterstützt von einigen im Linux-Kernel vertretenen Dateisystemen (ext4, f2fs). Arbeitet auf Verzeichnisebene und verschlüsselt einzelne Dateien
  - **dm-crypt** (LUKS): Technologie zur Verschlüsselung ganzer Partitionen, arbeitet auf Block-Ebene unterhalb des Dateisystems und kann daher mit allen Dateisystemen funktionieren, welche direkt auf ein Block-Medium benutzen (aber nicht Netzlaufwerke und virtuelle Laufwerke)

- **GoCryptFS**: Overlay-Verschlüsselung via FUSE (Filesystem-im-User-Space). Die Verschlüsselung geschied nicht im Linux-Kernel (=langsamer), kann auf beliebigen Verzeichnissen angewand werden (auch Netzlaufwerke)

Technologie	im Kernel	Geschwindigkeit	Scope	Netzlaufwerke
ext4 fscrypt	ja	++	Verzeichnisse	nein
dm-crypt (LUKS)	ja	++	Dateisystem / Partition	nein
gocryptfs	nein	–	Verzeichnisse	ja

2.24.1. ext4 mit Verschlüsselung

- Installation des Programms `fscrypt` zur Verwaltung von verschlüsselten Dateisystem-Onjekten:

```
# apt install fscrypt
```

- `fscrypt` initialisieren

```
# fscrypt setup
```

- Die `ext4` Funktion "Verschlüsselung" im Dateisystem auf `/dev/vda1` anschalten

```
# tune2fs -O encrypt /dev/vda1
```

- Einen neuen Schlüssel dem Linux-Session-Keyring hinzufügen und dem Verzeichnis `/encrypted/test` zuordnen

```
# mkdir -p /encrypted/test
# fscrypt encrypt /encrypted/test/
The following protector sources are available:
1 - Your login passphrase (pam_passphrase)
2 - A custom passphrase (custom_passphrase)
3 - A raw 256-bit key (raw_key)
Enter the source number for the new protector [2 - custom_passphrase]:
Enter a name for the new protector: villa
Enter custom passphrase for protector "villa":
Confirm passphrase:
"/encrypted/test/" is now encrypted, unlocked, and ready for use.
```

- Daten in das verschlüsselte Verzeichnis kopieren und die erweiterten BSD-Attribute der Dateien anzeigen. Es sollten die Attribute `---------E----e----` `/encrypted/test/passwd` erscheinen (E = encrypted, e = Datei mit "Extents", siehe auch `man chattr`)

```
# cp /etc/passwd /encrypted/test
# lsattr /encrypted/test
---------E----e------- /encrypted/test/passwd
```

- Dateisystem Schlüssel verwerfen (Dateien sind nicht mehr im Klartext sichtbar)

```
        # fscrypt lock /encrypted/test
        "/encrypted/test" is now locked.
    # ls -l /encrypted/test
        total 4
        -rw-r--r-- 1 root root 1684 Oct  1 14:17
2O2l3me829NsM_VbXeGiQaK5ANzWpC5stIA7XK7xIO3qAeskZ8CR_w
```

2.24.2. dmcrypt/LUKS

- LUKS Tools und LVM installieren

```
apt install cryptsetup lvm2
```

- Neues Physisches Volume in `/dev/sda5` erstellen

```
# pvcreate /dev/sda5
  Physical volume "/dev/sda5" successfully created.
```

- Eine neue Volumegroup erstellen

```
# vgcreate crypt_vg /dev/sda5
  Volume group "crypt_vg" successfully created
```

- ein neues logisches Volume im LVM erstellen (in der Volumegroup `crypt_vg`):

```
lvcreate --size 500M --name crypt crypt_vg
```

- Ein verschlüsseltes Laufwerk auf dem LV erstellen

```
 cryptsetup --verbose --verify-passphrase luksFormat \
     /dev/mapper/crypt_vg-crypt
```

- Das verschlüsselte Laufwerk im Linux-Device-Mapper anmelden

```
cryptsetup luksOpen /dev/mapper/crypt_vg-crypt crypt
```

- An dieser Stelle muss das neue Laufwerk nun im Device-Mapper auftauchen

```
ls -l /dev/mapper/crypt
```

- Das 'crypt' Laufwerk formatieren (hier ext4, andere Dateisysteme und SWAP sind möglich)

```
mkfs.ext4 /dev/mapper/crypt
```

- Ein leeres Verzeichnis anlegen und das neue Dateisystem dort mounten:

```
mkdir /crypt
mount /dev/mapper/crypt /crypt
```

- Das Dateisystem auf dem verschlüsselten Laufwerk mit Test-Daten füllen

```
 # cp -a /etc /crypt/
```

- Konfigurationsdatei für verschlüsselte Laufwerke anlegen

```
$EDITOR /etc/crypttab
```

- Eintrag für unser verschlüsseltes Laufwerk

```
crypt /dev/mapper/crypt_vg-crypt  none luks,timeout=180
```

- Eintrag in der `/etc/fstab`

```
/dev/mapper/crypt     /crypt    ext4    defaults  0 0
```

- Partition in der `/etc/fstab` eintragen, Reboot testen

2.24.3. dmcrypt/LUKS mit Container-Datei

- LUKS Tools und LVM installieren

```
apt install cryptsetup lvm2
```

- Container-Datei erstellen (hier 100MB)

```
# dd if=/dev/urandom of=geheim.img bs=100M count=1 iflag=fullblock
1+0 records in
1+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.736483 s, 142 MB/s
```

- Loopback-Gerät mit Container-Datei verbinden

```
# losetup /dev/loop0 geheim.img
```

- Ein verschlüsseltes Laufwerk auf dem Loopback-Gerät erstellen

```
# cryptsetup --verbose --verify-passphrase luksFormat /dev/loop0

WARNING!
========
This will overwrite data on /dev/loop0 irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /root/geheim.img:
Verify passphrase:
Key slot 0 created.
Command successful.
```

- Das verschlüsselte Laufwerk im Linux-Device-Mapper anmelden

```
cryptsetup luksOpen /dev/loop0 crypt
```

- An dieser Stelle muss das neue Laufwerk nun im Device-Mapper auftauchen

```
ls -l /dev/mapper/crypt
```

- Das 'crypt' Laufwerk formatieren (hier ext4, andere Dateisysteme und SWAP sind möglich)

```
mkfs.ext4 /dev/mapper/crypt
```

- Ein leeres Verzeichnis anlegen und das neue Dateisystem dort mounten:

```
mkdir /crypt
mount /dev/mapper/crypt /crypt
```

- Das Dateisystem auf dem verschlüsselten Laufwerk mit Test–Daten füllen

```
 # cp -a /etc /crypt/
```

- Konfigurationsdatei für verschlüsselte Laufwerke anlegen

```
$EDITOR /etc/crypttab
```

- Eintrag für ein verschlüsseltes Laufwerk in der `/etc/crypttab`

```
crypt /dev/mapper/crypt_vg-crypt  none luks,timeout=180
```

- Eintrag in der `/etc/fstab`

```
/dev/mapper/crypt     /crypt    ext4    defaults  0 0
```

2.24.4. gocryptfs

- gocryptfs verwendet eine dateibasierte Verschlüsselung, die als mountbares FUSE–Dateisystem implementiert ist. Jede Datei in gocryptfs wird als eine entsprechende verschlüsselte Datei auf der Festplatte gespeichert. Die verschlüsselten Dateien können in jedem Ordner auf der Festplatte, einem USB–Stick oder sogar im Netzwerklaufwerk (CIFS/Samba, NFS, Nextcloud, Dropbox etc) gespeichert werden.
- Ein Vorteil der dateibasierten Verschlüsselung gegenüber der Festplattenverschlüsselung (Blockverschlüsselung wie dm–crypt) besteht darin, dass verschlüsselte Dateien mit Standardtools wie Nextcloud–Desktop oder rsync effizient synchronisiert werden können. Außerdem ist die Größe des verschlüsselten Dateisystems dynamisch und nur durch den verfügbaren Speicherplatz begrenzt.
- goCryptFS Webseite: https://nuetzlich.net/gocryptfs/ [https://nuetzlich.net/gocryptfs/]

gocryptfs unter Debian einrichten

- gocryptfs installieren

```
apt install gocryptfs
```

- Verzeichnisse für die Klartext–Dateien und die verschlüsselten Dateien anlegen (Mountpoints)

```
mkdir ~/encrypted ~/plain
```

- Verzeichnis für die verschküsselten Daten vorbereiten

```
# gocryptfs -init ~/encrypted/
Choose a password for protecting your files.
Password:
```

```
Repeat:

Your master key is:

    5ea48963-d2eebeaf-3fefa893-82a98a7f-
    74b30c24-b3663b3d-acccce7c-63903952

If the gocryptfs.conf file becomes corrupted or you ever forget your password,
there is only one hope for recovery: The master key. Print it to a piece of
paper and store it in a drawer. This message is only printed once.
The gocryptfs filesystem has been created successfully.
You can now mount it using: gocryptfs /home/user/encrypted MOUNTPOINT
```

- gocryptfs Dateisystem unter `\~/plain` und `\~/encrypted` mounten

```
# gocryptfs ~/encrypted ~/plain
```

- Test des gocryptfs-Verzeichnisses

```
# cp /etc/passwd ~/plain/
# head -3 ~/plain/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin

# head -3 ~/encrypted/PAhZM_-LR7VDbOL71EW5hg | strings
tgBJx
u)TRb
JK((3a
?>Jb
#sCF
-.J4
]_7O
i"`{
lmC7H,
```

Verschlüsselte Backups mit "reverse mode"

- `gocryptfs` unterstützt einen speziellen "reverse Modus". Beim "normalen Modus" werden verschlüsselt gespeicherte Daten entschlüsselt im FUSE-Dateisystem angezeigt. Beim "reverse Mode" werden unverschlüsselte Daten im FUSE-Dateisystem verschlüsselt angezeigt.
    - Mit dem "reverse Mode" ist es einfach, von einem bestehenden Dateisystem (z.B. von Heimverzeichnis des Benutzers) ein verschlüsseltes Backup zu erstellen. Hierzu können bestehende Unix-Werkzeuge wie `tar`, `rsync` oder `cp` benutzt werden.

Dateisystem im "reverse-mode" mounten

- Heimverzeichnis für den Reverse-Mode vorbereiten

```
$ gocryptfs -reverse -init ~
Choose a password for protecting your files.
Password:
```

```
Repeat:

Your master key is:

    c9b93df2-90b1df59-9618bb1d-1a6e7379-
    643a322d-5a290479-5e3c742b-95c9df15

If the gocryptfs.conf file becomes corrupted or you ever forget your password,
there is only one hope for recovery: The master key. Print it to a piece of
paper and store it in a drawer. This message is only printed once.
The gocryptfs-reverse filesystem has been created successfully.
You can now mount it using: gocryptfs -reverse . MOUNTPOINT
```

- Backup-Verzeichnis erstellen

```
# mkdir /tmp/backup
```

- gocryptfs im "reverse Mode" anhängen

```
$ gocryptfs -reverse ~ /tmp/backup/
Password:
Decrypting master key
Filesystem mounted and ready.
```

- Verschlüsselung verifizieren

```
$ ls /tmp/backup/
58H2VYJoPM3R5yq9cmeMvXNj9x5rArx7MZu-hBNS3HY  TKIpQTQSwkS3pc2L9d3K9w
79s3miE-H0azjGqYgfRlOg                       TejQg8uPdSZYCTuhrKCa3g
7QN9ZQoS5wFgqtfUrqE4BQ                       XCpfaptlnRlOWiHyerj4iA
DDibcAU48C-C171E5llbxQ                       _JaUhWe5refqjc2YsCKEtg
DGCOM4xOMgYpnkkdkIng1A
et1cf2LT9Sc5A26hRWHLnX4Viybu_JccpBghUvtBEYE
HuKslsd8Jyt7vRmkw12Saw                       gocryptfs.conf
Ju4AJmmaI8HYyatd7G_vNA                       gocryptfs.diriv
MpyCCxDWYXMNU20P1o5S8Q                       iQzxl1-LPOQsJ0NR5gTpPQ
P80MPNYQy0SLJX8iKG1BvR7OBlFZtPtWmLPRiJkaMXA  xpQDM4E5oniygQbucmfyyQ
```

- Backup starten (Beispiel)

```
rsync -avi /tmp/backup user@backup-server.example:/backup
```

- Best practices:
  - Die Datei `gocryptfs.conf` nicht auf öffentlichen Servern sichern, es sei denn, die Daten sind durch ein sehr starkes Passwort geschützt.
  - Die Datei `gocryptfs.diriv` zusammen mit allen anderen verschlüsselten Dateien sichern. Diese Datei ist nicht geheim.

2.24.5. Links

- Unlocking LUKS2 volumes with TPM2, FIDO2, PKCS#11 Security Hardware on systemd 248
  [https://0pointer.net/blog/unlocking-luks2-volumes-with-tpm2-fido2-pkcs11-security-hardware-on-systemd-

248.html]

- GitHub – s-macke/CoverFS: Zero-knowledge client-side encrypted network filesystem [https://github.com/s-macke/CoverFS]
- Speeding up Linux disk encryption [https://blog.cloudflare.com/speeding-up-linux-disk-encryption/]
- `fscrypt` Quelldateien und Anleitung: https://github.com/google/fscrypt [https://github.com/google/fscrypt]

2.25. SECRET SHARING

- Shamir's Secret Sharing Scheme http://point-at-infinity.org/ssss/ [http://point-at-infinity.org/ssss/]
- ssss installieren

```
apt install ssss
```

- Ein Geheimniss (Passwort) auf 5 Shares aufteilen, bei denen 3 Shares reichen um das Geheimnis wiederherzustellen

```
ssss-split -t 3 -n 5
```

- Geheimnis aus 3 Shares wiederherstellen

```
ssss-combine -t 3
```

2.26. CLEVIS UND TANG

- Wird ein Linux-System mit per `dm-crypt` verschlüsseltem Dateisystem neu gebootet, so muss direkt nach dem Laden des Kernels und der Init-Ramdisk das Passwort zum entsperren des verschlüsselten Dateisystems eingegeben werden
  - Dies ist bei Laptop- und Desktop-Systemen möglich, bei Server-Systemen im Rechenzentrum aber sehr unpraktisch
  - Es gibt Anleitungen, um einen SSH-Dienst innerhalb der Initram-Umgebungs zu starten, und das EntschlÜsselungs-Passwort so über SSH einzugeben. Hier muss jedoch auch ein Mitglied des Administrationsteams zum Zeitpunkt des Neustarts aktiv werden□—□ungünstig für automatisierte Neustarts (z.B. nach zeitkrtischen Sicherheitsupdates)
- Die Linux-Komponenten *Clevis* (auf dem Server mit verschlüsseltem Dateisystem, in der Init-Ramdisk) und *Tang* (Schlüsselserver im Netz) kann das Entschlüsselungspasswort auf mehrere Server im Netz aufgeteilt werden (per Shamirs Secret Sharing)
  - Beim Boot erfragt die *Clevis*-Komponente direkt nach dem Start des Linux-Kernels das Entschlüsselungspasswort von den *Tang*-Servern
  - Diese Server sollten nicht (alle) im öffentlichen Internet erreichbar sein□—□die Entschlüsselung der Datenträger sollte nur im Rechenzentrum funktionieren. Werden die Datenträger aus dem Rechenzentrum entfernt, sind nicht mehr alle *Tang*-Server erreichbar und somit kann der Schlüssel nicht vollständig zusammengesetzt werden. Dies schützt vor Diebstal von Datenträgern im Rechnenzentrum
- Anleitung "Tang/Clevis for a luks-encrypted Debian Server": https://www.ogselfhosting.com/index.php/2023/12/25/tang-clevis-for-a-luks-encrypted-debian-server/ [https://www.ogselfhosting.com/index.php/2023/12/25/tang-clevis-for-a-luks-encrypted-debian-server/]

CHAPTER 3. TAG 3/4 (NOVEMBER)

3.1. CVE (COMMON VULNERABILITIES AND EXPOSURES)

- CVE = eindeutige Kennzeichnung von Sicherheitslücken in Software
- Aufbau: CVE-<Jahr>-<Nummer>
- CVE Datenbank https://web.nvd.nist.gov/view/vuln/search [https://web.nvd.nist.gov/view/vuln/search]

3.1.1. Common Vulnerability Scoring System (CVSS)

- Spezifikation: https://www.first.org/cvss/specification-document [https://www.first.org/cvss/specification-document]

CVSS V2

- Access Vector (AV) – wie kann der Angreifer die Lücke ausnutzen
  - AV:L lokaler Angriff möglich
  - AV:A Angriff aus den gleichen Subnetz möglich
  - AV:N Remote über das Netzwerk
- Access Complexity (AC) – wie schwierig ist es für einen Angreifer die Lücke auszunutzen, sobald er im System ist
  - AC:H Hoch
  - AC:M Medium
  - AC:L Niedrig
- Authentication (Au) – wie oft muss sich der Angreifer am System anmelden/authentisieren muss
  - Au:M mehrfache Authentisierung notwendig (Benutzer → sudo Root)
  - Au:S einmalige Anmeldung genügt (normaler Benutzer)
  - Au:N garnicht
- Confidentiality Impact © – Auswirkung auf die Datensicherheit
  - C:N keine
  - C:P Teil-Daten
  - C:C alle Daten
- Integrity Impact (I) – Auswirkung auf die Integrität des Systems
  - I:N keine Auswirkung
  - I:P System teilweise nicht mehr vertrauenswürdig
  - I:C System nicht mehr vertrauenswürdig
- Availability Impact (A) – Auswirkung auf die Verfügbarkeit des Systems
  - A:N keine Auswirkung

    - A:P teilweise beeinträchtigung der Verfügbarkeit
    - A:C Komplettausfall des Systems

CVSS V3

- Angriffs Vektor (AV)
    - Network (N) – Angriff über Netzwerk möglich
    - Adjacent (A) – Angreifer muss im lokalen Netz sein
    - Local (L) – Angreifer muss lokal auf dem System sein
    - Physical (P) – Angreifer muss physischen Zugang zu dem System haben
- Angriffs Komplexität (AC)
    - Low (L) – keine besondern Kentnisse beim Angreifer erforderlich
    - High (H) – der Angreifer braucht spezielle Kenntnisse
- benötigte Privilegien (PR)
    - None (N) – keine besonderen Privilegien notwendig
    - Low (L) – Normale Benutzeranmeldung notwendig
    - High (H) – Angreifer benötigt spezielle Rechte
- Unterstützung duch einen Benutzer (UI)
    - None (N) – der Angriff kann ohne Aktion eines lokalen Benutzers erforderlich sein
    - Required ® – der Angriff benötigt die Unterstützung eines lokalen Benutzers
- Scope (Reichweite) (S)
    - Unchanged (U) – die Sicherheitslücke beeinträchtigt nur das angegriffene System
    - Changed © – über die Sicherheitslücke werden auch andere Systeme als das angegriffene beeinträchtigt
- Auswirkungen auf den Schutz von Daten (Confidentiality Impact) ©
    - High (H) – Datenschutz kann nicht mehr sichergestellt werden
    - Low (L) – einige Daten können in die Hände des Angreifers fallen
    - None (N) – die Sicherheitslücke hat keine Auswirkungen auf den Schutz der Daten
- Auswirkungen auf die Integrität der Daten (I)
    - High (H) – die Daten können vom Angreifer verändert werden und der Angreifer kann damit direkt Schaden verursachen
    - Low (L) – der Angreifer kann Daten verändern, hat aber keine Kontrolle über die Auswirkungen
    - None (N) – der Angreifer kann keine Daten verändern
- Auswirkungen auf die Verfügbarkeit des Systems (A)
    - High (H) – das System fällt aus

    - Low (L) – es gibt Beeinträchtigungen der Verfügbarkeit
    - None (N) – die Sicherheitslücke hat keine Auswirkungen auf die Verfügbarkeit
- Verfügbarkeit von Angriffscode
    - Undefiniert (X) – es liegen keine Informationen über die Verfügbarkeit von Schadcode vor
    - High (H) – es gibt automatisierte Angriffs–Programme
    - Functional (F) – es existiert Schadcode der manuell ausgeführt werden kann und in den meisten Fällen erfolgreich ist
    - Proof–of–Concept (P) – es gibt Beispiele für Schadcode, dieser ist jedoch auf dem meisten Systemen nicht einsetzbar
    - Unproven (U) – es gibt noch keinen Code oder die Sicherheitslücke ist nur theoretisch
- Abhilfe
    - Undefiniert (X) – es liegen noch keine Informationen über Patche vor
    - Unavailable (U) – es gibt noch keinen Schutz gegen die Sicherheitslücke
    - Workaround (W) – es gibt keinen Fix, aber eine Anleitung die Ausnutzung der Sicherheitslücke zu verhindern
    - temporary fix (T) – es gibt einen vorläufigen Patch
    - official fix (O) – es gibt einen offiziellen Patch
- Glaubwürdigkeit
    - Undefiniert (X) – es können keine Aussagen zur Glaubwürdigkeit der Meldung gemacht werden
    - Confirmed © – die Sicherheitslücke wurde bestätigt
    - Reasonable ® – die Sicherheitslücke ist glaubhaft
    - Unknown (U) – es gibt Berichte über Sicherheitsprobleme, aber eine Verbindung zur Sicherheitslücke konnte noch nicht bestätigt werden

3.1.2. EUVID

- Die Europäische CVE–Datenbank, betreut von der ENISA, ist eine Alternative zur US–CVE Datenbank https://euvd.enisa.europa.eu/ [https://euvd.enisa.europa.eu/]

3.1.3. OpenCVE

- OpenCVE = OpenSource Vulnerability Management Platform
    - Eine Plattform zur Filterung und Verwaltung von Sicherheitsmeldungen in Unternehmen und Organisationen
- https://www.opencve.io/ [https://www.opencve.io/], Source: https://github.com/opencve/opencve [https://github.com/opencve/opencve]

3.2. INCIDENCE RESPONSE

- ISO/IEC 27035-1:2016 Information security incident management — Part 1: Principles of incident management https://www.iso.org/obp/ui/#iso:std:iso-iec:27035:-1:ed-1:v1:en:sec:A [https://www.iso.org/obp/ui/#iso:std:iso-iec:27035:-1:ed-1:v1:en:sec:A]
- ISO/IEC 27035-2:2016 Information security incident management — Part 2: Guidelines to plan and prepare for incident response https://www.iso.org/obp/ui/#iso:std:iso-iec:27035:-2:ed-1:v1:en:sec:A [https://www.iso.org/obp/ui/#iso:std:iso-iec:27035:-2:ed-1:v1:en:sec:A]
- Atlassian Incident Handbook https://pages.eml.atlassian.com/rs/594-ATC-127/images/Atlassian-incident-management-handbook-.pdf [https://pages.eml.atlassian.com/rs/594-ATC-127/images/Atlassian-incident-management-handbook-.pdf]
- ENISA "Good Practice Guide for Incident Management" https://www.enisa.europa.eu/sites/default/files/publications/Incident_Management_guide.pdf [https://www.enisa.europa.eu/sites/default/files/publications/Incident_Management_guide.pdf]
- NIST "Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile" https://csrc.nist.gov/pubs/sp/800/61/r3/final [https://csrc.nist.gov/pubs/sp/800/61/r3/final]
- CISA "Cybersecurity Incident & Vulnerability Response Playbooks" https://www.cisa.gov/sites/default/files/2024-08/Federal_Government_Cybersecurity_Incident_and_Vulnerability_Response_Playbooks_508C.pdf [https://www.cisa.gov/sites/default/files/2024-08/Federal_Government_Cybersecurity_Incident_and_Vulnerability_Response_Playbooks_508C.pdf]
- Centre for Cyber Security Belgium: "CYBER SECURITY INCIDENT MANAGEMENT GUIDE" https://cybersecuritycoalition.be/wp-content/uploads/cybersecurity-incident-management-guide_EN.pdf [https://cybersecuritycoalition.be/wp-content/uploads/cybersecurity-incident-management-guide_EN.pdf]
- BSI: Vorfallunterstützung https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Cyber-Sicherheitslage/Reaktion/Vorfallunterstuetzung/MIRT/mirt_node.html [https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Cyber-Sicherheitslage/Reaktion/Vorfallunterstuetzung/MIRT/mirt_node.html]
- BSI: Sicherheitsvorfall https://www.bsi.bund.de/DE/IT-Sicherheitsvorfall/Unternehmen/unternehmen_node.html [https://www.bsi.bund.de/DE/IT-Sicherheitsvorfall/Unternehmen/unternehmen_node.html]
- BSI: Erste Hilfe bei einem schweren ITSicherheitsvorfall https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Cyber-Sicherheit/Themen/Ransomware_Erste-Hilfe-IT-Sicherheitsvorfall.pdf [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Cyber-Sicherheit/Themen/Ransomware_Erste-Hilfe-IT-Sicherheitsvorfall.pdf]

3.2.1. Vorbereitungen auf den Notfall

- Personen und Rollen für die Bearbeitung eines Sicherheitsvorfalls benennen, Zuständigkeiten schriftlich dokumentieren.
- Kontaktadressen von Forensikern/Strafverfolgungsbehörden besorgen und bereithalten.
- Datei `security.txt` auf im Internet sichtbaren Servern ausrollen und aktuell halten (https://securitytxt.org/ [https://securitytxt.org/], RFC 9116 "A File Format to Aid in Security Vulnerability Disclosure" [https://www.rfc-editor.org/rfc/rfc9116]).

- PGP/GPG/SMIME Schlüssel erstellen und aktuell halten. E–Mail–Verschlüsselung regelmässig benutzen, um Übung im Umgang mit verschlüsselter E–Mail zubehalten.
- Alternative Kommunikationswege (Mastodon, externes E–Mail–System, externer Web–Server) vorbereiten. Kunden über diese alternativen Kommunikationswege informieren!
- Forensik Live–Linux–CD–ROM/USB–Stick bereithalten und regelmässig üben (1/2 jährlich).
- Erstellen von Platten–Images üben.
- Ersatz–Hardware vorhalten (Festplatten, SSD), Speicherplatz für Platten–Images reservieren und getrennt von der übrigen IT Infrastruktur bereitstellen.
- Beutel für manipulationssichere Beweissicherung bestellen und "vor Ort" bereithalten (z.B. Rechnerraum, Rechenzentrum).
- Hardware–RAID meiden (Alternativen: Software RAID – DM–RAID, btrfs, zfs). Hardware–RAID macht die forensische Untersuchung von Festplatten schwierig, da die Forensiker über die gleiche RAID–Hardware verfügen müssen (= Zeitverlust im Notfall).
- Minimale–Installationen benutzen. Minimale Installationen des Betriebssystems und der Anwendungen verringern die Anzahl von Dateien welche bei einem Sicherheitsvorfall untersucht werden müssen.
- Datensparsamkeit – nur Daten, die nicht gespeichert werden, können nicht mißbraucht werden → Datenhaltung kostet.
- *Incident–Modus* für die IT–Systeme definieren –→ welche Teile der IT können im Fall eines Angriffs ohne starke Auswirkungen auf den Betrieb ausgeschaltet oder abgeschottet werden?
- *Incident–Modus* automatisiert per Configuration–Management System anschalten –→ Testen
- Bei erkannten Angriffen 4 Augen–Prinzip beachten, nicht alleine arbeiten!

3.2.2. Kunden/Benutzer im Falle eines Sicherheitsvorfalls informieren

- Wenn die eigene Kommunikationsstruktur (z.B. E–Mail) betroffen ist, die alternativen Kommunikationswege benutzen (Mastodon, extern gehostete Website), Atlassian Statuspage https://www.atlassian.com/software/statuspage [https://www.atlassian.com/software/statuspage] etc)
- Dokumentierte Notfall–Prozedur zur Kunden–Kommunikation vorhalten und vorab an das Incident–Team verteilen

3.2.3. Spurensicherung

- VM Snapshots erstellen (per GPG signieren)
- Von einem Forensik–Linux booten (Kali–Linux, Deft o.ä.)
- Revisionssichere Images (dcfldd, aff4) erstellen http://www.osforensics.com/tools/create–disk–images.html [http://www.osforensics.com/tools/create–disk–images.html]
- Images sofort nach der Erstellung per GPG signieren (mit externer Signaturdatei)
- Erst Platten–Images erstellen, dann System analysieren (Log–Dateien etc. anschauen).
- Festplatten ausbauen und durch fabrikneue Platten ersetzen, Festplatten mit kompromittiertem Daten versiegeln (lassen)
- Analyse des Einbruches auf den Read–Only–Dateisystem–Images (nicht auf den echten Platten,

die echten Platten sind ein Beweismittel und dürfen nach der Versiegelung nicht angefasst werden)

3.2.4. Backup/Rebuild eines kompromitierten Systems

- Server komplett neu aufsetzen (Docker container via `dockerfile` neu bauen, Foreman/Ansible/Salt etc, VM Templates)
- Festplatte(n) austauschen
- Nur Daten ohne ausführbare Teile (Maschinencode, Java, .NET, Python pyc, PHP, Scripte etc) zurückspielen
  - Bei der Einrichtung der Server schon auf eine Trennung von System, Anwendungen und Daten achten!
- Scripte neu erstellen, von einem bekannt **sauberen** Backup zurückspielen oder einzeln per Pair–Review (2 Personen Review) prüfen

3.3. INTRUSION DETECTION

3.3.1. rhash

- `rhash` ist ein Werkzeug zum schnellen Erstellen einer Hash–Datenbank von Dateien in einem Linux–System
- `rhash` eignet sich als *Ad–Hoc* Werkzeug beim arbeiten an einem Incident
- rhash Installation

```
apt install rhash
```

- Datenbank mit den Hash–Werten erstellen (Hash=SHA3/Keccak mit 224 bit, Datenbankformat wie bei BSD)

```
rhash -r --sha3-224 --bsd /etc -o hash.lst
```

- Datenbank anschauen

```
less hash.lst
```

- Datenbank sichern (in Produktionsumgebung)

```
scp hash.lst benutzer@sicherer-server.example.com
```

- Dateien gegen die Datenbank prüfen

```
rhash -r -c --bsd hash.lst
```

- Die Benutzer–Authentisierungsdateien durch Anlegen eines neuen Benutzers ändern

```
adduser intruder
```

- nochmal Dateien gegen die Datenbank prüfen

```
rhash -r -c --bsd hash.lst
```

- Nur die "nicht-OK" Dateien anzeigen

```
rhash --skip-ok -r -c --bsd hash.lst
```

3.3.2. AIDE – Advanced Intrusion Detection Engine

- AIDE ist eine populäre Software um Einbrüche in Linux-Systemen zu erkennen
  - AIDE erstellt eine Datenbank mit kryptografischen Hashes der Dateien eines Linux-Systems
  - Nach Änderungen am System (z.B. Einspielen von Updates, Änderungen an den Konfigurationsdateien) muss die Datenbank neu aufgebaut werden
  - AIDE ist eine Implementierung der Idee hinter der kommerziellen Software "Tripwire"
  - AIDE benötigt vergleichsweise viel CPU und Hauptspeicher
  - AIDE sichert in der Standard-Konfiguration Dateihashes mittels vieler verschiede Hash-Algorithmen. Dies verbraucht viel Zeit und CPU-Leistung. Es wird daher empfohlen die Anzahl Hashes auf ein bis zwei zu beschränken.

Installation

- Installation von AIDE

```
apt install aide aide-common
```

- Konfiguration anschauen und ggf. ändern

```
$EDITOR /etc/aide/aide.conf
```

Initiale Datenbank anlegen

- AIDE Datenbank erstellen (Dauer ca. 3-6 Minuten auf den Linuxhotel Laptops)

```
aideinit
```

- Datenbank sichern (in Produktionsumgebungen, nicht im Training)

```
scp  /var/lib/aide/aide.db.new.gz user@secure-machine
```

- Eine unbefugte Änderung am System simulieren

```
groupadd intruders
```

System gegen die Datenbank prüfen

- AIDE Check laufen lassen

```
aide --check -c /etc/aide/aide.conf
```

- Ausgabe des AIDE-Checks

```
Start timestamp: 2025-11-21 11:16:45 +0000 (AIDE 0.19.1)
AIDE found differences between database and filesystem!!
Ignored e2fs attributes: EINV

Summary:
  Total number of entries:      94108
  Added entries:                0
  Removed entries:              0
  Changed entries:              9

---------------------------------------------------
Changed entries:
---------------------------------------------------

d >....    . .. .  : /dev/char
f >.... mci.H.. .  : /etc/group
f >.... mc..H.. .  : /etc/group-
f >.... mci.H.. .  : /etc/gshadow
f >.... mc..H.. .  : /etc/gshadow-
d =.... mc.. .. .  : /run/rdnssd
f =.... mci.... .  : /run/rdnssd/resolv.conf
f =.... mc..... .  : /run/resolvconf/run-lock
f >b... mc..H.. .  : /var/log/aide/aideinit.log

---------------------------------------------------
Detailed information about changes:
---------------------------------------------------

Directory: /dev/char
 Size      : 2900                            | 2920

File: /etc/group
 Size      : 828                             | 846
 Mtime     : 2025-11-21 11:02:34 +0000       | 2025-11-21 11:16:38 +0000
 Ctime     : 2025-11-21 11:02:34 +0000       | 2025-11-21 11:16:38 +0000
 Inode     : 5149                            | 3572
 SHA256    : AsNFSYdZn8kHe+molxWSGcXC9xbDSBaX | 3WrkiSJSZ35xzTLLcohWvyA43M3Z2nSf
             GTXRvB8oqHs=                    | p2hOs4ZjngQ=
 SHA512    : n7y7oxi+NF0Q3U7jOaky3REtHeATcq5Z | 5HcBb4x906Lcw1yK7ErIOeoSVCDIhpb0
             7v3eh6x5kmhDKxRvtSzvy7NPfhO6zlWU | SPhNPsOy97IQyvePhHBvIoqIKi83MMUR
             3t2yWp1FNxA043KaBgAk5A==        | 411T3dPBy6HjhK+LUmMTbA==
 STRIBOG256: UF8SIV8W2RQ1EoIrKQgUSeFbDOVkGGMJ | e9XK7FDJ2oaEzkcqGwEu2jthxsc88p9E
             HE3vK6T6CP8=                    | jDlTlJXX7O4=
 STRIBOG512: BsJDdWnEarQAwFMs8IhAxZPc3fs4xcSd | qOcJL/V2lYq02unTJuTQJaS5fWHc6x8T
             4BNchAwfnQwn0VWJ47LajxphcrP2PWHy | GnIEtXGU4dN21n5J5LJYFV8hhXMkvTJF
             A2UdUDctBDpalYHI+wq3lA==        | q5oWh3Z8PEpYmHwz0DvfXA==
 SHA512/256: 8nNE7/O/xqqI8oFOjRvHmkOJSbZvKzg7 | 3KhR1YyUGtcKs7mSHp52G0J1Dm/as3D5
             2MRiU8J1kAk=                    | /cx0+Q6VVhI=
 SHA3-256  : 3jvMRLAvjb04brpW4DcCIEnmhoBfPhix | +Amgxgu50PzzsKjL2CKIjNaS97uPryKi
             kKYp4nl2o8Q=                    | l2voFiyhkuE=
 SHA3-512  : sGBWaiPB3hPlMF8MPwq0cr3Y1QpJ2Vru | 1MsGo8U4IJ8OE6YMWtFMWnSxfXOBBRSK
             iiZ9+RWJxdaU9UK+bHJvvKl5dNg7TTqP | FMyWRLY1PZUgImvkdnrvlXx09aQnTgj9
             2S0Q8IxsNPGFRBhwwu9Vfw==          | 8SLkyPiNEfavCFyW8d0a/Q==
```

```
[...]
```

3.3.3. Samhain

- Samhain ist ein host–basierte Intrusion Detection System
  - Neben Änderungen an wichtigen Systemdateien überwacht Samhain das System auf ungewöhnliche Meldungen in den Log–Dateien, versteckten Prozessen, neuen geöffneten Netzwerk–Ports und vieles mehr
  - Dabei ist Samhain recht sparsam mit den Systemressourcen (Speicher, CPU)
  - Samhain ist Open–Source (Programmierer: Dr. Rainer Wichmann)
  - mehrere Server mit Samhain können mittels einer zentralen Web–Oberfläche (Beltane) überwacht und verwaltet werden. Beltane ist kommerzielle Software und nicht Open–Source.

Samhain Installation

- Installation

```
apt install samhain
```

- Samhain internen Schlüssel ergänzen (<key> sollte durch eine lange — 20 Zeichen oder mehr — zufällige Zahl ersetzt werden)

```
systemctl stop samhain
samhain --add-key=<key>@/usr/sbin/samhain
cp /usr/sbin/samhain.out /usr/sbin/samhain
rm -f /var/lib/samhain/samhain_file
samhain -t init
systemctl start samhain
```

- Samhain Konfigurationsdatei anschauen/anpassen.

```
$EDITOR /etc/samhain/samhainrc
```

- Datenbank auf einen sicheren Server kopieren (in Produktionsumgebungen)

```
scp /var/lib/samhain/samhain_file nutzer@secure-host:.
```

- Samhain ist als Service–Dämon gestartet

```
systemctl status samhain
```

- Test Passwort–Dateien ändern

```
# test, Datei (/etc/shadow und /etc/passwd) ändern
passwd user
```

- Check mit Ausgaben nur Level "crit"

```
samhain -t check --foreground -l none -p crit
```

- Samhain Datenbank aktualisieren

```
samhain -t update --foreground
```

3.4. LYNIS

- Lynis ist ein Sicherheitsscanner welcher veraltete Software-Versionen und bekannt unsichere Konfigurationen in Linux-Installationen aufspürt
- Homepage: https://cisofy.com/lynis/ [https://cisofy.com/lynis/]
- Lynis installieren (die **lynis** Version in den Debian Paket-Repositories ist veraltet)

```
 curl -fsSL https://packages.cisofy.com/keys/cisofy-software-public.key | sudo gpg
--dearmor -o /etc/apt/trusted.gpg.d/cisofy-software-public.gpg
 echo "deb [arch=amd64,arm64 signed-by=/etc/apt/trusted.gpg.d/cisofy-software-
public.gpg] https://packages.cisofy.com/community/lynis/deb/ stable main" | sudo tee
/etc/apt/sources.list.d/cisofy-lynis.list
 apt update
 apt install lynis
```

- Lynis Version Information abfragen

```
 lynis update info
```

- Lynis starten

```
 lynis audit system
```

- Lynis Informationen anzeigen

```
lynis show help
lynis show commands
lynis show settings
lynis show profiles
lynis --man-page
```

- In der Produktion: Lynis als Cron-Job (oder Systemd-Timer-Unit) eingerichten und periodisch ausführen, Ausgaben/Reports per E-Mail versenden

3.5. DEBIAN DIENSTE NICHT SOFORT NACH DER INSTALLATION STARTEN

- Datei `/etc/systemd/system-preset/00-disable-all.preset` anlegen mit dem Inhalt

```
disable *
```

- Unter `/usr/sbin/policyd-rc.d` ein Shell Script erzeugen, welches Fehler #101 zurückliefert

```
# echo -e '#!/bin/bash\nexit 101' > /usr/sbin/policy-rc.d
# chmod +x /usr/sbin/policy-rc.d
# /usr/sbin/policy-rc.d
# echo $?
101
```

- Link: https://major.io/2016/05/05/preventing-ubuntu-16-04-starting-daemons-package-installed/ [https://major.io/2016/05/05/preventing-ubuntu-16-04-starting-daemons-package-installed/]

3.6. APT UPDATES AUTOMATISIEREN

- Nach der Installation von Debian sind die automatischen Updates ausgeschaltet. Die Konfiguration der automatischen Updates findet sich unter `/etc/apt/apt.conf.d/50unattended`:

```
apt install unattended-upgrades apt-listchanges
$EDITOR /etc/apt/apt.conf.d/50unattended-upgrades
----
Unattended-Upgrade::Mail "root";
Unattended-Upgrade::Automatic-Reboot-Time "02:00";
Unattended-Upgrade::Automatic-Reboot "false";
Unattended-Upgrade::Automatic-Reboot-WithUsers "true";
```

- automatische Updates können über den Schalter `APT::Periodic::Unattended-Upgrade` in der Datei `/etc/apt/apt.conf.d/20auto-upgrades` an- bzw. ausgeschaltet werden.

3.7. NAMESPACES

- Dieses Kapitel zeigt, das Namespaces (und damit Container) eine Standard-Technologie des Linux-Kernel ist und keiner weiteren Software benötigt. `systemd-nspawn` und auch **Docker** sind *nur* Verwaltungsprogramme für die Namespaces und Controll-Groups des Linux-Kernels.
- Namepspaces 'virtualisieren' die Sicht auf Ressourcen des Linux-Kernels
- Programme wie Docker, Podman, systemd-nspawn, LXC/LXD, Firejail, Google-Chrome Browser etc. benutzen die Namespaces im Linux-Kernel
- Verfügbare Namespaces in Linux

Namespace	Konstante	Isolation
Cgroup	CLONE_NEWCGROUP	Cgroup root Verzeichnis
IPC	CLONE_NEWIPC	System V IPC, POSIX message queues
Network	CLONE_NEWNET	Network devices, stacks, ports, etc.
Mount	CLONE_NEWNS	Mount points
PID	CLONE_NEWPID	Process IDs
User	CLONE_NEWUSER	Benutzer und Gruppen IDs
UTS	CLONE_NEWUTS	Hostname und NIS Domain Name

3.7.1. Namespace Beispielprogramm

- mit diesem kleinen Beispielprogramm wird gezeigt, das Linux-Container mit nur einem Systemaufruf erzeugt werden.
- Quelle: https://blog.lizzie.io/linux-containers-in-500-loc.html [https://blog.lizzie.io/linux-containers-in-500-loc.html]
- C-Compiler und Entwicklungs-Tools installieren

```
apt install build-essential
```

- Unser Ausgangs-C-Programm: Leeres Verzeichnis anlegen und die leere Quelltextdatei im Editor öffnen

```
cd /usr/src
mkdir namespaces
cd namespaces
$EDITOR namespaces.c
----
#define _GNU_SOURCE
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <sched.h>
#include <signal.h>
#include <unistd.h>

#define STACK_SIZE (1024 * 1024)

static char child_stack[STACK_SIZE];
char* const child_args[] = {
  "/bin/bash",
  NULL
};

int child_main(void* arg)
{
  printf("Namespace aktiv\n");
  execv(child_args[0], child_args);
  printf("Ooops\n");
  return 1;
}

int main()
{
  printf("Namespace wird erzeugt\n");
  int child_pid = clone(child_main, child_stack+STACK_SIZE, \
   CLONE_NEWUTS | CLONE_NEWIPC | SIGCHLD, NULL);

  if (child_pid < 0) {
    printf("Es konnte kein namespace erzeugt werden (Fehler %d)\n", child_pid);
    return 1;
  }
```

```
  waitpid(child_pid, NULL, 0);
  printf("Namespace beendet\n");
  return 0;
}

----
```

- Programm übersetzen

```
gcc -o namespaces namespaces.c
```

- Programm ausführen

```
./namespaces
```

- Den Namespace mit `exit` verlassen, dann einen Process-Namespace zum Programm hinzufügen

```
...
 int child_pid = clone(child_main, child_stack+STACK_SIZE, \
   CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWPID | SIGCHLD, NULL);
...
```

- Neu übersetzen und ausführen (im Namespace das Proc-Dateisystem neu mounten um die Prozess-Isolierung via `top` oder `ps` zu sehen: `mount -t proc proc /proc`

```
gcc -o namespaces namespaces.c
```

- Namespace via `exit` beenden und das `proc` Dateisystem auf dem Host neu mounten `mount -t proc proc /proc`
- Einen Netzwerk-Namespace hinzufügen

```
...
  int child_pid = clone(child_main, child_stack+STACK_SIZE, \
   CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWPID |  CLONE_NEWNET | SIGCHLD, NULL);
...
```

- Die Netzwerkconfiguration im Namespace mit `ip` oder `ifconfig` anschauen

3.8. NETZWERK NAMESPACES PER `IP` KOMMANDO

- Netzwerk Namespace **netns1** erzeugen

```
# ip netns add netns1
# ip netns list
```

- Befehl im Netzwerk-Namespace ausführen

```
# ip netns exec netns1 ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

- Loopback ist "down", ein `ping` geht nicht

```
# ip netns exec netns1 ping 127.0.0.1
connect: Network is unreachable
```

- Nun bringen wir das Loopback–Interface "up"

```
# ip netns exec netns1 ip link set dev lo up
# ip netns exec netns1 ping 127.0.0.1
```

- Wir erstellen nun **virtuelle** Netzwerkschnittstellen

```
# ip link add veth0 type veth peer name veth1
# ip link
```

- Netzwerkschnittstelle **veth1** wird in den Netzwerk–Namespace **netns1** verschoben

```
# ip link set veth1 netns netns1  # auch physische NICs koennen in Namespaces
verschoben werden
# ip link
```

- Wir konfigurieren eine IP–Adresse auf **veth1** im Namespace

```
# ip netns exec netns1 ip address add 10.1.1.1/24 dev veth1
# ip netns exec netns1 ip a
# ping 10.1.1.1  # ping in den Namespace geht jetzt noch nicht!
```

- Eine IP–Adresse auf dem **veth0** auf dem Linux–Host konfigurieren

```
# ip addr add 10.1.1.2/24 dev veth0
# ip link set dev veth0 up
```

- Ping von aussen in den Namespace geht jetzt!

```
# ping 10.1.1.1
```

- Ping aus den Namespace nach **draussen** sollte auch gehen

```
# ip netns exec netns1 ping 10.1.1.2
```

- der Namespace hat eine eigene Routing–Tabelle und eine eigene Netfilter–Firewall

```
ip netns exec netns1 route
ip netns exec netns1 ip route show
ip netns exec netns1 iptables -L
```

- Namespace wieder löschen

```
ip netns delete netns1
```

3.9. EINFACHE CONTAINER MIT `UNSHARE`

- Mit dem Programm `unshare` können die einzelnen Namespaces separat erstellt und eingeschaltet werden. Wird kein zu startendes Programm angegeben, so wird die Standard–Shell des Systems mit den neuen Namespaces gestartet. Beispiel: ein Namespace mit privatem Netzwerk, Mount– und Prozess–ID– Namespace:

```
    # unshare -n -p -f --mount-proc
```

- Container mit Benutzer `root`, welcher nicht `root` auf dem Host ist

```
   sudo unshare --map-root-user --user sh
```

3.10. NAMESPACES AUFLISTEN

- Befehl lsns:

```
[root@centos-sec-oct-2017 ~]# lsns
        NS TYPE  NPROCS   PID USER   COMMAND
4026531836 pid       81     1 root   /usr/lib/systemd/systemd --system --deserialize 22
4026531837 user      81     1 root   /usr/lib/systemd/systemd --system --deserialize 22
4026531838 uts       81     1 root   /usr/lib/systemd/systemd --system --deserialize 22
4026531839 ipc       81     1 root   /usr/lib/systemd/systemd --system --deserialize 22
4026531840 mnt       78     1 root   /usr/lib/systemd/systemd --system --deserialize 22
4026531856 mnt        1    12 root   kdevtmpfs
4026531956 net       81     1 root   /usr/lib/systemd/systemd --system --deserialize 22
4026532212 mnt        1 11299 root   /usr/lib/systemd/systemd-udevd
4026532227 mnt        1   551 chrony /usr/sbin/chronyd
```

3.11. CONTAINER/NAMESPACE MIT SYSTEMD

- Jedes Linux mit Systemd bietet mächtige Container-Verwaltungs-Werkzeuge mit
- `systemd-nspawn` arbeitet neben Image-Dateien für Container auch mit installieren Linux-Root-Dateien in einem beliebigen Verzeichnis auf dem Container-Host-System
    - Damit ist es sehr einfach, ein Container-System aufzusetzen und Dateien zwischen dem Container-Host und dem Linux im Container auszutauschen
- In diesem Kapitel installieren wir als Beispiel ein Rocky-Linux (eine freie Red Hat Enterprise Linux Distribution) in einem Verzeichnis unter Debian
    - Andere Linux-Distributionen wie Ubuntu, Suse, Arch-Linux oder ältere Debian-Versionen wären auch möglich
    - Wichig ist das die Dateien im Container-Verzeichnis für die CPU-Architektur des Container-Hosts übersetzt wurden und die Programme keinen neueren Kernel benötigen (keine neuen System-Calls)
- Seit Debian 9 müssen die Systemd-Container-Tools separat installiert werden:

```
apt install systemd-container
```

- Wir erstellen ein Verzeichnis für den neuen Container

```
mkdir -p /srv/container/rocky-linux9
```

- Initialisieren eine neue RPM-Instanz im Container-Verzeichnis (bei Debian mit **debboostrap**, bei SUSE mit **RPM** und **zypper**)

```
apt -y install rpm dnf
rpm --root /srv/container/rocky-linux9 --initdb
```

- Das Basis-Paket für Rocky-Linux laden (Achtung: URLs ändern sich regelmäßig)

```
wget https://ftp.plusline.net/rockylinux/9/BaseOS/x86_64/os/Packages/r/rocky-release-9.6-1.3.el9.noarch.rpm
wget https://ftp.plusline.net/rockylinux/9/BaseOS/x86_64/os/Packages/r/rocky-repos-9.6-1.3.el9.noarch.rpm
rpm --nodeps --root /srv/container/rocky-linux9 -ivh rocky-release-9.6-1.3.el9.noarch.rpm rocky-repos-9.6-1.3.el9.noarch.rpm
```

- Das Rocky-Linux Basis-System installieren

```
dnf -y --nogpg --releasever=9 --installroot=/srv/container/rocky-linux9 \
  install systemd passwd dnf procps-ng iproute tmux rocky-gpg-keys
echo 9 > /srv/container/rocky-linux9/etc/dnf/vars/releasever
```

- Eine Shell im Container starten

```
systemd-nspawn -D  /srv/container/rocky-linux9
```

- Root-Passwort setzen und neuen Benutzer anlegen

```
-bash-4.2$ passwd
-bash-4.2$ adduser user
-bash-4.2$ passwd user
```

- Shell im Container verlassen

```
-bash-4.2# exit
```

- Container booten

```
systemd-nspawn -bD  /srv/container/rocky-linux9
```

- Weitere Software im Container installieren (hier den Apache Webserver)

```
dnf install httpd
```

- Anwendung im Rocky Linux Container starten (prüfen das kein andere Prozess auf dem Container-Host die Ports 80 und/oder 443 belegt)

```
systemctl enable --now httpd
```

- Der Apache Webserver innerhalb des Rocky Linux Containers sollte nun vom Firefox des Debian unter http://localhost [http://localhost] erreichbar sein.
- Container stoppen (im Container eingeben)

```
container# poweroff
```

- Container mit privatem Netzwerk-Namespace starten:

```
systemd-nspawn -bD  /srv/container/rocky-linux9 --private-network
```

- Container können auch mit virtuellen Netzwerkkarten und/oder Port-Mappings konfiguriert werden

- Systemd–Befehle um einen Container vom Host aus zu kontrollieren

```
machinectl list
machinectl status rocky-linux9
machinectl terminate rocky-linux9
machinectl kill rocky-linux9
```

- Per **nsenter** mit dem Container verbinden (es wird eine Prozess–ID eines Container–Prozesses benötigt!)

```
nsenter -m -u -i -n -p -t <pid-im-container> /bin/bash
```

- Beispiel (auf Basis des im Rocky–Linux laufenden Apache2 Prozesses)

```
nsenter -m -u -i -n -p -t $(pgrep -o httpd) /bin/bash
```

3.12. FIREJAIL

- Firejail ist eine leichtgewichtige Sandbox für Linux–Programme auf Basis von Linux–Namespaces, secomp–bpf und der Linux–Firewall
  - Firejail wurde ursprünglich für die Absicherung des Firefox–Browsers erstellt, kann heuter auch für die Absicherung vieler andere Programme benutzt werden
  - Der Fokus von Firejail liegt auf Desktop–Linux–Programmen, jedoch kann Firejail auf auch Linux–Servern eingesetzt werden
- Homepage https://firejail.wordpress.com/ [https://firejail.wordpress.com/]
- Quellcode https://github.com/netblue30/firejail [https://github.com/netblue30/firejail]
- Moderne Versionen von Firejail unterstützen die Absicherung der DNS Namensauflösung mittels DNS–over–HTTPS und DNS–over–TLS mit dem FDNS Projekt https://firejaildns.wordpress.com/ [https://firejaildns.wordpress.com/]
- Firejail installieren

```
  apt install firejail firejail-profiles
```

- Den Konsolen–Webbrowser `links` installieren (als Beispielprogramm)

```
# apt install links
```

- Einen unprivilegierten Benutzer (z.B. `user`, `gabi` etc., ggf. den Benutzer neu anlegen) in der Datei `/etc/firejail/firejail.users` eintragen (Die Datei ggf. anlegen, pro Benutzer eine Zeile), um die Benutzung von `firejail` durch diesen Benutzer zu erlauben
- Zum unprivilegierten Benutzer wechseln

```
# su - user
```

- Für den Benutzer das Verzeichnis `.ssh` (SSH Schlüssel und Konfiguration) erstellen. Firejail sichert dieses Verzeichnis gesondert ab, so das Programme ohne spezielle Firejail–Rechte (wie das `ssh` Programm) auf die Inhalte dieses Verzeichnisses **nicht** zugreifen können

```
$ cd ~
$ mkdir .ssh
```

```
$ chmod 700 .ssh
```

- Test von Firejail: wir starten eine Bash–Shell kontrolliert von Firejail

```
# firejail bash
```

- Teste die folgenden Aktionen in der Firejail–Bash:
  - Prozessliste auflisten
  - Programm `top` starten
  - Zeige die Datei `.bash_history` an (wenn die Datei noch nicht existiert dann die Bash einmal verlassen und wieder neu mit `firejail bash` starten)
  - Wechsle in das Verzeichnis `.ssh`
  - Liste alle Dateien im Heimverzeichnis mit `ls -la` und vergleiche mit der Ausgabe ohne Firejail. Was fällt auf?
- Firejail `bash` verlassen
- Firejail mit einem Browser (hier Links)
  - Den Browser `links` im Jail starten (als unprivilegierter Benutzer)

```
# su - user
$ firejail links https://notes.defaultroutes.de
```

- Auf der Webseite des Trainings die PDF–Datei `LinuxSecurityEinfuehrung.pdf` finden, auswählen und per Taste `d` (Download) sichern, danach den Browser `links` mit der Taste `q` verlassen
  - Wo ist die Datei gespeichert worden?
    - Erstelle im Heimverzeichnis des Benutzers ein Verzeichnis mit den Namen `Downloads`

```
$ mkdir ~/Downloads
```

- Starte `links` nochmals via `firejail`, lade das PDF von der Trainings–Webseite und speichere es im Verzeichnis `Dowloads`. Nach dem Beenden des Browsers ist die Datei dort zu finden
- Starte den Browser mit einer Datei–System–URL des Heimverzeichnisses `links file://.` mit und ohne FireJail. Welche Unterschiede gibt es?
- Erstelle (als Benutzer `root`) einen symbolischen Link von `/usr/bin/firejail` zu `/usr/local/bin/links`
  - Wird Firejail über einen solchen Symlink aufgerufen, so startet es das Programm mit dem Namen unter der Kontrolle von Firejail. Der Benutzer kann daher das Programm "direkt" aufrufen, ohne den Befehl `firejail` voranstellen zu müssen

```
$ ln -s /usr/bin/firejail /usr/local/bin/links
```

- Firejail kann automatisiert für unterstützte Programme solche Wrapper–Links installieren. Die Wrapper–Links überlagern die Programm–Namen von schützenswerten Linux–Anwendungen.

```
$ sudo firecfg
```

- Firejail Programme überwachen (in einem separaten Terminal starten und dann z.B. `links` starten)

```
firejail --top
```

- Im System verfügbare Firejail Profile auflisten

```
ls /etc/firejail
```

3.13. DOCKER/PODMAN

- Podman ist eine "drop-in" (Kommandozeilen-Kompatible) Alternative zu Docker (Podman wird von RedHat als Open Source Projekt vorangetrieben). Die weiteren Ausführungen in diesen Kapitel gelten bis auf wenige Ausnahmen auch für Podman.

3.13.1. Docker als Sicherheits-Werkzeug

- Docker bietet gegenüber dem Betrieb von klassischen Linux-Installationen (auf „bare Metal“ oder virtualisiert) einige Vorteile:
  - Anwendungen werden voneinander isoliert. Selbst die Bestandteile einer Anwendung (z.B. LAMP – Linux/Apache/MySQL/PHP) können in eigene Docker-Container ausgelagert und voneinander isoliert werden.
  - die Rechte einer Anwendung im Docker-Container kann per Seccomp-bpf und Linux-Capabilities beschränkt werden
  - die Sicht auf den Netzwerkstack kann für Anwendungen eingeschränkt werden (Admin-Netzwerk für Anwendungen nicht sichtbar)
  - Docker unterstützt ein Implementations-Design, bei dem Anwendungs-Programme und Daten getrennt gehalten werden (Anwendungen in Container-Images, Daten in Speicher-Volumes ausserhalb der Container)
  - Über Dockerfiles sind die Container jederzeit reproduzierbar. Bei einem Einbruch in einen Docker-Container kann dieser zu Beweiszwecken eingefroren werden und durch ein neues, sauberes Container-Image ersetzt werden
  - Docker-Container werden nicht per Software-Update aktualisiert, sondern durch ein neues Container-Image mit aktueller Software ersetzt. Hierdurch kann sich ein (auch bisher unbemerkter) Einbrecher nicht im System festsetzen
- Nachteile:
  - die Isolierung von Anwendungen durch Linux-Container-Virtualisierung ist nicht so stark und vollständig wie bei einer Voll-Virtualisierung (VMWare, HyperV, VirtualBox). Linux-Container können jedoch innerhalb einer Voll-Virtualisierung eingesetzt werden und somit werden die Vorteile beider Systeme genutzt
  - Anwendungs-Installationen müssen angepasst werden, um Programme und Daten sauber zu trennen
  - Docker erhöht die Komplexität der Installation
  - Docker-Verwaltungswerkzeuge (Kubernetes etc) erhöhen die Komplexität und haben ggf. eigene Sicherheitsprobleme

3.13.2. Docker Installieren

- Docker installieren und starten

```
apt install docker.io
usermod -aG docker <username>
```

- Docker testen

```
docker run hello-world
```

3.13.3. Erste Schritte mit Docker

- Docker Image herunterladen

```
# docker pull almalinux
# docker images
```

- Ein Alma-Linux (eine weitere freie Distribution von Red Hat Enterprise Linux) mit Bash starten

```
# docker run --name=application1 -it almalinux /bin/bash
docker# dnf update -y
docker# dnf install -y procps httpd
docker# dnf clean all
docker# exit
# docker ps -a
```

- Mit laufendem Docker Container verbinden

```
docker attach application1
```

- Docker Container Statistiken anzeigen

```
# docker top <container>
# docker stats <container1> <container2> ...
```

- Einen weiteren Prozess im Docker Container ausfuehren

```
# docker exec -d <container> touch /etc/new-config
# docker exec -t -i <container> /bin/sh
```

- Docker Container stoppen

```
# docker stop application1
```

- Docker bei Applikations-Ende automatisch neu starten

```
docker run --restart=always
docker run --restart=on-failure:5
```

- Docker Container Konfiguration anzeigen → siehe Go Template wie man nach Konfiguration-Informationen filtern kann https://golang.org/pkg/text/template/ [https://golang.org/pkg/text/template/]

```
docker inspect <container>
```

```
docker inspect --format '{{ .NetworkSettings.IPAddress }}' <container>
```

- Docker Container im Hintergrund (`-d`) starten

```
docker run --name <name> -d <image> <command>
```

- Ausgaben der Anwendung im Docker Container anschauen

```
docker logs <container>
docker logs -f <container>
docker logs --tail -f <container>
docker logs --tail 50 <container>
```

- Docker Log-Ausgaben können mittels eigener *log drivers* auf andere Ziele umgelenkt werden(json-file, syslog, none, ...)

```
docker run --log-driver="syslog" ...
```

- Ein Docker Image aus einem Dockerfile (Docker-Image Bauanleitung) erstellen

```
mkdir -p docker-work
cd docker-work
$EDITOR Dockerfile
-----
# Debian with nginx
# Version 1
FROM debian:latest

MAINTAINER Linuxhotel Trainer

# Update image
RUN apt update
# Add httpd package. procps and iproute are only added to investigate the image later.
RUN apt -y install nginx
RUN echo container.example.com > /etc/hostname

# Create an index.html file
RUN bash -c 'echo "Your Web server test on Debian Docker Container is successful." >>
/var/www/html/index.html'
# create a nginx start-command
CMD ["nginx", "-g", "daemon off;"]
------
docker build -t debian-nginx --no-cache .
docker run -d -t --name=deb-nginx1 -p 8080:80 debian-nginx
```

3.13.4. Docker Container verbinden

- Volumes

```
docker run -v <local-dir>:<container-dir>[:ro] ...
```

- Volumes im Dockerfile definieren. Hierdurch werden Volumes beim Erstellen des Containers ausserhalb des Containers erstellt (unter `/var/lib/docker/...` auf dem Linux-Host) und in den

Container eingebunden

```
# Dockerfile
[...]
VOLUME /data
VOLUME /var/www/html
```

3.13.5. Docker Übung: Caddy-Webserver:

- installiere ein Image mit dem 'Caddy' Webserver (https://caddyserver.com [https://caddyserver.com]) aus der Docker Hub Registry (abiosoft/caddy). Caddy ist ein in der Sprache `go` geschriebener Webserver, welcher automatisch TLS/x509 Zertifikate von Let's Encrypt besorgt und aktualisiert.
- starte einen Container aus dem Image, verbinde Port 2015 vom Container auf einen freien Port auf dem Host
- teste die Caddy-Webseite mit dem Firefox auf dem Laptop http://<ip-des-Laptop>:<port> [http://<ip-des-Laptop>:<port>]
- starte eine Shell im Caddy-Webserver Container per `docker exec`. Suche das Verzeichnis mit den HTML-Dateien.
    - stoppe den Container, lösche den Container (`docker rm`)
    - erstelle ein leeres Verzeichnis auf dem Container-Host (Laptop) unter `/srv/websites/html`. Erstelle eine einfache HTML-Datei `index.html` in diesem Verzeichnis.
    - starte einen neuen Container, verbinde das Verzeichnis `/srv/websites/html` vom Container-Host in den Container (Parameter `-v`, siehe oben).
    - Teste den Webserver mit dem Firefox Browser
    - Ändere die Datei `index.html` unter `/srv/websites/html`, teste das die Änderungen im Browser sichtbar sind
    - erstelle einen neuen NGINX Container (auf Basis des NGINX Dockerfiles oben in der Anleitung), bei dem das Verzeichnis `/srv/websites/html` in den Webroot des NGINX-Webservers gemappt wird (**Achtung:** anderen Port als für den Caddy-Webserver benutzen. Beide Webserver-Container sollten zur gleichen Zeit aktiv sind)
    - Einige Befehle zum Probieren (als Benutzer `root` auf dem Host)

```
docker logs <container>   # Anzeige der Logausgaben des Hauptprozesses
docker top <container>    # Anzeige der Prozesse im Container
docker stats <container>  # Anzeige der vom Container benutzten Resourcen
```

Beispiellösung

- Caddy-Webserver Image aus dem Docker Repository

```
docker run --name=caddyweb -d -p 8088:2015 abiosoft/caddy
```

- Im Docker-Container nach den HTML-Dateien suchen

```
docker exec caddyweb find / -name index.html
```

- Caddy Container stoppen und löschen

```
docker stop caddyweb
docker rm caddyweb
```

- Verzeichnis für die Webseiten anlegen und eine `index.html` Datei erstellen

```
mkdir -p /srv/websites/html
echo "Dies ist meine Webseite" > /srv/websites/html/index.html
```

- Caddy Container mit dem externen Webseiten-Verzeichnis starten

```
docker run --name=caddyweb -v /srv/websites/html:/srv -d -p 8088:2015 abiosoft/caddy
```

3.13.6. mit Docker Images arbeiten

- Docker Image History anschauen

```
docker images
docker history <image-name>
```

- Docker Image(s) in eine Datei sichern. Metadaten und Schichten bleiben erhalten

```
docker save -o mydockerapp.tar <image-name> [<image-name2>]
```

- Docker-Image aus einer Datei laden (welche mit `docker save` erstellt wurde)

```
docker load -i mydockerapp.tar
```

- Docker *Container* exportieren (nur das Dateisystem wird exportiert, die Schichten/Layer verschwinden)

```
docker ps -a
docker export -o mydockerapp.tar <container-name>
xz -v mydockerapp.tar
```

- Docker Container vom Tarball importieren

```
xzcat mydockerapp.tar.xz | docker import - <image-name>
```

- Docker Container in ein Image umwandeln (z.B. für forensische Analyse)

```
docker commit <container-ID> <image-name>
```

3.13.7. lokale Docker Registry

- Docker Registry als Docker-Container starten. Die Docker Registry benutzt Port 5000/tcp.

```
docker run -d -p 5000:5000 registry
```

- Ein Docker Image "taggen"

```
docker images
docker tag <images-id> localhost:5000/debian-nginx:01
```

- Image in die Registry laden (push)

```
docker push localhost:5000/debian-nginx:01
```

- Docker Image lokal löschen (Achtung: Docker Images können erst gelöscht werden, wenn keine aktiven under inaktiven Container dieses Image benuzten. Ggf. müssen erst die auf diese Images basierenden Container per docker rm <container> gelöscht werden)

```
docker rmi localhost:5000/debian-nginx:01
docker rmi debian:latest
docker images
```

- Image aus der lokalen Registy (neu) laden

```
docker pull localhost:5000/debian-nginx:01
```

- wenn die Kommunikation zwischen Registry und Docker-Client nicht über die Loopback-Schnittstellen läuft (nicht localhost), dann muss diese Kommunikation mit TLS/SSL abgesichert sein. Dazu muss auf der Registry ein x509 Zertifikat installiert werden. Optionen:
  - von Let's Encrypt ein x509 Zertifikat besorgen und einbauen. Da Let's Encrypt Zertifkate nur 3 Monate gültig sind, sollte dieser Prozess automatisiert und überwacht sein
  - ein x509 Zertifikat von einer Zertifizierungsstelle (CA) kaufen
  - Option ohne TLS: per SSH einen Tunnel von Port 5000 des Clients auf Port 5000 des Registry-Hosts aufbauen und den Docker-Client gegen den „localhost" Port laufen lassen. Bonus: per Systemd-Socket-Activation auf dem Client kann der SSH-Tunnel automatisch aufgebaut werden.
- Liste der Images in einer lokalen Registry laden und anzeigen

```
curl -X GET http://localhost:5000/v2/_catalog | python -m json.tool
```

3.14. DOCKER SICHERHEIT

3.14.1. Docker Images

- Top ten most popular docker images each contain at least 30 vulnerabilities (Februar 2019): https://snyk.io/blog/top-ten-most-popular-docker-images-each-contain-at-least-30-vulnerabilities/ [https://snyk.io/blog/top-ten-most-popular-docker-images-each-contain-at-least-30-vulnerabilities/]

3.14.2. Container Sicherheit

- keine SUID Programme im Container
- keinen "echten" Benutzer "root" (UID0) im Container (Capabilities benutzen)
- Anwendungen im Container nicht unter einem Root-Account betreiben
- Capabilities sparsam einsetzen (kein CAP_SYS_ADMIN)
- Dateien in `/tmp` sind keine *Daten*, nicht von extern mounten
- Programme (im Container) und Daten (ausserhalb des Containers) trennen

- sensitive Programme read-only in den Container mappen/mounten
- Container auf bekannte Sicherheitslücken der im Container benutzen Software überwachen
- Container oft und regelmässig auffrischen

3.14.3. Audit-Regeln für Docker einrichten

- Beispiel-Regelwerk (generisch, muss ggf. für die eigene Repository angepasst werden)

```
 -w /etc/docker -k docker
-w /etc/docker/key.json -k docker
-w /etc/docker/daemon.json -k docker
-w /etc/docker/certs.d -k docker
-w /etc/sysconfig/docker -k docker
-w /etc/sysconfig/docker-network -k docker
-w /etc/sysconfig/docker-storage -k docker
-w /etc/sysconfig/docker-storage-setup -k docker
-w /usr/lib/systemd/system/docker.service -k docker
-w /usr/bin/docker-containerd -k docker
-w /usr/bin/docker-containerd-current -k docker
-w /usr/bin/docker-containerd-shim -k docker
-w /usr/bin/docker-containerd-shim-current -k docker
-w /usr/bin/docker -k docker
-w /usr/bin/docker-current -k docker
-w /usr/bin/docker-ctr-current -k docker
-w /usr/bin/dockerd -k docker
-w /usr/bin/dockerd-current -k docker
-w /usr/bin/container-storage-setup -k docker
-w /var/lib/docker -k docker
-w /var/run/docker.sock -k docker
```

3.14.4. Docker-Engine nicht auf einem (ungeschützten) TCP Port betreiben

3.14.5. Docker Repository mit TLS absichern (x509 Zertifikate benötigt)

- in der Datei `/etc/sysconfig/docker`

```
dockerd --tlsverify ...
```

3.14.6. User-Namespaces verwenden

- in der Datei `/etc/sysconfig/docker`

```
dockerd --userns-remap=default ...
```

3.14.7. Legacy-Registry abschalten

- in der Datei `/etc/sysconfig/docker`

```
dockerd --disable-legacy-registry ...
```

3.14.8. Ulimits auf Container setzen

- in der Datei `/etc/sysconfig/docker`, Format `--default-ulimit <ulimit-spec>=<soft-limit>:<hard-limit>`
- Information ueber ULIMIT Schluesselwoerter in `/etc/security/limits.conf`
- Default-Ulimit setzen

```
dockerd --default-ulimit nofile=50:150 --default-ulimit nproc=10:20
```

- Ulimit fuer einzelnen Container

```
docker run --ulimit nofile=20:50 ...
```

3.14.9. Docker Container nicht als Benutzer root betreiben

```
    docker run -u dockeruser -d ...
```

3.14.10. Capabilities von Container-Prozessen beschränken

- Docker Container mit beschraenkten Capabilities starten

```
docker run --cap-drop=all --cap-add [notwendige capabilities] ...
```

- Capabilities eines Prozesses in einem Docker Container herausfinden am Beispiel des nginx Webservers

```
 # Prozess-ID des nginx Prozesses herausfinden
docker exec $(docker ps -q) pgrep -a nginx
# Capabilities der Prozess-ID 1 (nginx) abfragen
docker exec $(docker ps -q) cat /proc/1/status | grep CapEff | awk '{print $NF}'
00000000a80425fb
# Capabilties dekodieren und Docker-Parameter ausgeben
capsh --decode=00000000a80425fb | echo "--cap-drop=all --cap-add={$(sed -e 's/.*=//' -e
's/cap_//g')}"
--cap-drop=all --cap
-add={chown,dac_override,fowner,fsetid,kill,setgid,setuid,setpcap,net_bind_service,net_
raw,sys_chroot,mknod,audit_write,setfcap}
```

- Die Capabilities unter --cap-add pruefen und ggf. einschraenken

3.14.11. /etc/localtime im Container Read-only

- die Datei /etc/localtime im Container Read-Only setzen, verhindert, das die Zeitzone im Container veraendert wird

```
docker run -v /etc/localtime:/etc/localtime:ro
```

3.14.12. Speicher- und CPU-Resources des Containers einschränken

Docker Parameter	Beschreibung
–cpu–period	Laenge der CPU CFS (Completely Fair Scheduler) Zeitscheibe
–cpu–quota	CPU CFS (Completely Fair Scheduler) quota
–cpuset–cpus	Container an CPUs binden (0–3, 0,1).
–cpuset–mems	Container an Speicher–Nodes binden (0–3, 0,1) nur fuer NUMA Maschinen
–kernel–memory	Kernel Speicher–Limit
–m, –memory	Speicher Limit
–memory–reservation	Limit fuer Speicherreservierungen (Virtueller noch nicht benutzer Speicher)
–memory–swap	Vollstaendiges Speicherlimit (memory + swap), '–1' schaltet virtuellen Speicher (SWAP) aus

3.14.13. Dateisysteme "nur–lesbar" einbinden

- Das Root–Dateisystem im Container „nur–lesend“ einbinden

```
docker run --read-only ...
```

- Container Volume nur–lesend einbinden

```
docker run -v /volume:ro ...
```

- Pfade des Hosts nur–lesend in den Container einbinden

```
docker run -v /srv/container/name:/srv:ro ...
```

3.14.14. Fähigkeiten eines Docker Containers per SELinux einschränken

Für Linux–Sicherheits–Module (LSM) auf Label Basis (Role Based Access Control, RBAC) kann der Benutzer, die Rolle, der Tyoe und der Level für einen Container festgelegt werden. LSM funktionieren nicht innerhalb von Containers, sondern nur für einen gesamten Container.

```
    docker run --security-opt=[label=user:USER]
```

3.14.15. Fähigkeiten eines Docker Containers per AppArmor einschränken

- Docker mit AppArmor Profilen benutzen https://cloud.google.com/container–optimized–os/docs/how–to/secure–apparmor [https://cloud.google.com/container–optimized–os/docs/how–to/secure–

apparmor]

- Docker Default AppArmor Profil: https://docs.docker.com/engine/security/apparmor/ [https://docs.docker.com/engine/security/apparmor/] https://github.com/docker/labs/tree/master/security/apparmor [https://github.com/docker/labs/tree/master/security/apparmor]

3.14.16. Syscalls einschränken mit Seccomp

- LWN A seccomp overview https://lwn.net/Articles/656307/ [https://lwn.net/Articles/656307/]
- Default Docker Seccomp Profile https://github.com/moby/moby/blob/master/profiles/seccomp/default.json [https://github.com/moby/moby/blob/master/profiles/seccomp/default.json]
- Linux x86_64 syscall Tabelle https://filippo.io/linux-syscall-table/ [https://filippo.io/linux-syscall-table/]

```
man syscalls
```

- Docker Container mit eigenem Seccomp Profil starten:

```
docker run --security-opt=[seccomp=/pfad/zur/eignen/seccomo-profile.json]
```

3.14.17. Vorsicht bei dem Einsatz von KSM (Kernel-Same-Pages-Merging)

- Gefahr eines Rowhammer Angriffs auf Speicherbereiche über VM/Containergrenzen https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_tuning_and_optimization_guide/chap-ksm [https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_tuning_and_optimization_guide/chap-ksm]

```
# systemctl status ksm
● ksm.service - Kernel Samepage Merging
   Loaded: loaded (/usr/lib/systemd/system/ksm.service; enabled; vendor preset:
enabled)
   Active: active (exited) since Mi 2017-10-25 19:20:35 CEST; 13h ago
  Process: 777 ExecStart=/usr/libexec/ksmctl start (code=exited, status=0/SUCCESS)
 Main PID: 777 (code=exited, status=0/SUCCESS)
    Tasks: 0
   Memory: 0B
   CGroup: /system.slice/ksm.service

Okt 25 19:20:35 notebook51 systemd[1]: Starting Kernel Samepage Merging...
Okt 25 19:20:35 notebook51 systemd[1]: Started Kernel Samepage Merging.
```

- Rowhammer Angriff: https://de.wikipedia.org/wiki/Rowhammer [https://de.wikipedia.org/wiki/Rowhammer]
- Attacking a co-hosted VM: A hacker, a hammer and two memory modules https://thisissecurity.stormshield.com/2017/10/19/attacking-co-hosted-vm-hacker-hammer-two-memory-modules/ [https://thisissecurity.stormshield.com/2017/10/19/attacking-co-hosted-vm-hacker-hammer-two-memory-modules/]

- KSM ausschalten (mehrere Optionen)

```
# /usr/libexec/ksmctl stop
# echo 0 > /sys/kernel/mm/ksm/run
# systemctl disable ksm
# systemctl disable ksmtuned
# echo 2 > /sys/kernel/mm/ksm/run
```

3.15. WEITERE CONTAINER-TOOLS

3.15.1. Das mbox Programm

- das **mbox** Programm bietet eine einfache, simple **sandbox** mittels CGroups und Namespaces. **mbox** kann benutzt werden um die Ausführung von unbekannten Programmen (Scripts, Binärprogramme) einzuschränken und zu überwachen https://github.com/tsgates/mbox [https://github.com/tsgates/mbox]

3.15.2. Minijail

- https://github.com/omegaup/minijail [https://github.com/omegaup/minijail]
- https://lwn.net/Articles/700557/ [https://lwn.net/Articles/700557/]

3.15.3. Bocker

- Docker nachimplementiert in 100 Zeilen Bash
- https://github.com/p8952/bocker [https://github.com/p8952/bocker]

3.16. SSH

3.16.1. SSH Fingerprint eines Servers prüfen

- Bei der ersten Verbindung zu einem neuen SSH-Server wird der Fingerprint des Servers angezeigt
  - Um "Machine-in-the-Middle" (MitM) Angriffe zu vermeiden sollte dieser Fingerabdruck geprüft werden
- SSH Fingerprint eines Schlüssels auf einem Server anzeigen (zur Prüfung eines ersten Logins)

```
ssh-keygen -l -f /etc/ssh/ssh_host_<algorithm>_key.pub
```

3.16.2. SSH-Schlüssel Fingerprints in DNS

- Alternativ zur manuellen Prüfung der Server-Fingerabdrücke können diese im DNS abgelegt werden
  - SSH-Clients können den Fingerabdruck aus dem DNS laden und automatisch prüfen
- Beispiel SSHFP-Records

```
# dig sshfp smtp3.strotmann.de
smtp3.strotmann.de.     1800    IN      SSHFP   1 1
```

```
BE915F185EE68A333BDF3B700BB1259A690D2D62
smtp3.strotmann.de.     1800    IN      SSHFP   3 2
C791A1F8A0AD41D3F85FF9D30E5CDFFB8821D513E7BEE8AD328DF65F 3D07D25C
smtp3.strotmann.de.     1800    IN      SSHFP   3 1
F545E655EF7C6F0B3328C611352AD822A7F5B419
smtp3.strotmann.de.     1800    IN      SSHFP   1 2
AFCFB9EE88E5C0AE6108061AFC8DBD9ED39301FB2DF2D77E674693F3 59BADE00
```

- SSH Fingerprint im DNS hinterlegen (DNSSEC Absicherung empohlen!). SSHFP Records erstellen (auf dem Server, je ein Aufruf pro Schlüssel-Type (RSA, ECDSA, ED25519 ...):

```
ssh-keygen -r <hostname> -f /etc/ssh/ssh_host_<algorithm>_key.pub
```

- SSH-Client für die Prüfung des SSH-Fingerabdrucks in DNS konfigurieren

```
ssh -o "VerifyHostKeyDNS ask" <hostname>
```

- in SSH-Config Datei übernehmen

```
$EDITOR ~/.ssh/config
----
HOST *
  VerifyHostKeyDNS ask
```

- Test ohne SSHFP Prüfung

```
ssh host.example.com
The authenticity of host 'host.example.com (2001:db8:2b6:0:ba27:ebff:fe12:30db)' can't
be established.
ECDSA key fingerprint is SHA256:ue1FlRUMmkPNhgFE55yqnnFl58FOlMnDGheCxQn2tWg.
Matching host key fingerprint found in DNS.
ECDSA key fingerprint is MD5:2e:a1:3c:94:d0:cb:ed:85:67:2e:7a:85:1c:bc:f3:70.
Matching host key fingerprint found in DNS.
Are you sure you want to continue connecting (yes/no)?
```

- mit `VerifyHostKeyDNS yes` gesetzt

```
ssh -v <host>
[...]
debug1: found 4 secure fingerprints in DNS
debug1: matching host key fingerprint found in DNS
[...]
```

- Wenn OpenSSH Probleme hat, die DNSSEC Authentizität der SSHFP zu prüfen, muss ggf. EDNS in der Resolver-Konfiguration des Linux eingeschaltet werden (der OpenSSH-Client verifiziert SSHFP-Records im DNS via DNSSEC, und die DNSSEC-Signaturen sind oft grösser 512 Byte und bedingen die EDNS(0)-Erweiterung des DNS welche DNS-Antworten bis 4K erlauben). Die übrige DNS-Infrastruktur muss auch DNSSEC agnostisch sein, d.h. DNSSEC Daten unverändert durchleiten.

```
$EDITOR /etc/resolv.conf
----
[...]
options edns0
```

3.16.3. SSH Known Hosts hashen

- In vielen SSH-Installationen werden die IP-Adressen oder Domain-Namen der Ziel-Server in der Datei `known_hosts` in Klartext abgelegt
  - Dies ist problematisch wenn die SSH-Konfiguration von einem Angreifer geklaut wird: der Angreifer weiss daher sofort auf welchen Server die geklauten privaten Schlüssel benutzt werden können
- Bestehende `known_hosts` Datei hashen

```
ssh-keygen -H -f .ssh/known_hosts
rm .ssh/known_hosts.old
```

- SSH-Client-Konfiguration anpassen, so das neue Einträge gehashed werden. In der Datei `.ssh/config`

```
[...]
HashKnownHosts yes
[...]
```

3.16.4. Übung: SSH mit Schlüsseln statt mit Passwort (ca. 20 Minuten)

- in dieser Übung arbeiten wir als Benutzer und als Administrator. Lege mit dem Übungs-Partner die Rolle fest (Benutzer oder Administrator) und führe die Übung durch. Danach wechselt die Rolle.
- Administrator: lege einen Benutzeraccount für den Kollegen/Kollegin vor/hinter Dir im Kurs an (Passwort und Benutzername absprechen)
- Administrator: OpenSSH Server installieren (wenn nicht schon geschehen)

```
sudo apt install openssh-server
```

- Benutzer: Teste aus, dass Du Dich auf dem System des Übungs-Partners einloggen kannst. Bei der ersten Kontaktaufnahme wird der Fingerprint des Servers angezeigt. Dieser Fingerprint kann über den Administrator des Servers angefragt werden.

```
user: ssh user@serverXX.dane.onl
```

- Benutzer: nach dem Test wieder aus dem fernen Rechner ausloggen
- Benutzer: als normaler Benutzer (`user`, nicht `root`) einen neuen SSH-Schlüssel erstellen (4096 RSA). Eine Passphrase vergeben.

```
user$ ssh-keygen -b 4096
```

- Benutzer: den öffentlichen Schlüssel per SSH auf den fernen Rechner übertragen:

```
user$ ssh-copy-id user@serverYY.dane.onl
```

- Es empfiehlt sich, die Schlüssel mit `ssh-copy-id -i .ssh/...` zu begrenzen
- Benutzer: Alternativ die Datei `\~/.ssh/id_rsa.pub` vom eigenen Rechner auf den fernen Rechner übertragen (z.B. per E-Mail senden) und den Administrator bitten, den eigenen Schlüssel in die Datei `.ssh/authorized_keys` einzutragen.

- Administrator: Nun sind die Schlüssel ausgetauscht, und wir können Anmeldung per Passwort auf dem Server–System abschalten. Als `root` Benutzer:

Passwort–Authentisierung abschalten

Die Benutzung von Passwörter für die SSH–Authentisierung ist einfach aber liefert nur eine minimale Sicherheit. Passwörter sind oft zu erraten oder per Brute–Force–Angriff ermittelbar. Besser sind kryptografische Schlüssel. SSH bietet die Authentisierung per Schlüssel, jedoch wird die Passwort–Anmeldung als Rückfall–Mechanismus benutzt. Um SSH richtig abzusichern sollte daher die Passwort–Anmeldung in der Konfiguration des SSH–Servers abgeschaltet werden.

```
$EDITOR /etc/ssh/sshd_config
------
PermitRootLogin no
PubkeyAuthentication yes
PasswordAuthentication no
------
```

- Administrator: SSH–Dienst neu starten

```
systemctl restart sshd
```

- Benutzer: Anmeldung am fernen Rechner testen, dies sollte nun nach Eingabe der Passphrase funktionieren

```
user$ ssh user@serverYY
```

- Benutzer: Wenn ein Benutzer ohne Schlüssel versucht, sich am fernen Rechner anzumelden, wird er gleich abgewiesen

```
ssh root@serverYY
```

- Info Benutzer: die Fingerprints der Server werden beim Client unter `\~/.ssh/known_hosts` gespeichert
- Info Administrator: die öffentlichen Schlüssel werden auf dem fernen Rechner in der Datei `\~/.ssh/authorized_keys` gespeichert

3.16.5. SSH Passphrase ändern

- Die Passphrase auf einem privaten SSH–Schlüssel kann belienig geändert werden:

```
ssh-keygen -p -f .ssh/id_ed25519
```

3.16.6. SSH–Agent

- Der SSH–Agent speichert die Passphrases der Schlüssel auf dem Client im Hauptspeicher. Hierdurch muss die Passphrase für Schlüssel nicht bei jeder Verbindung neu eingegeben werden.
- SSH–Agent starten (wird normalerweise in der Benutzer–Sitzung gestartet, z.B. über ".profile" oder `systemd --user`)

```
#entweder in der aktiven Shell
eval $(ssh-agent)
```

```
#oder als prefix-Befehl
ssh-agent tmux #oder bash
```

- Die privaten SSH-Schlüssel dem SSH-Agent hinzufügen

```
ssh-add
ssh-add <Schlüssel-Datei>
```

- Fingerprints aller Schlüssel im Agent anzeigen

```
ssh-add -l
```

- Alle Schlüssel im Agent anzeigen

```
ssh-add -L
```

- Agent sperren/entsperren

```
ssh-add -x
ssh-add -X
```

- Schlüssel im SSH-Agent löschen

```
ssh-agent -D
```

3.16.7. Agent-Forwarding

- Agent-Funktion an einem Jumphost weitergeben. Nun kann sich der Benutzer vom Jumphost mit Schlüssel an weiteren SSH-Servern anmelden

```
ssh -A <jumphost>
```

Übung: SSH-Forwarding

- Der Nachbar sollte uns einen Benutzer auf seinem System mit Deinem Benutzernamen erstellen
- Für diese Übung müssen wir unseren öffentlichen Schlüssel auf den Server eines Nachbarn installieren (dies wurde ggf. schon vorher im Training gemacht: Übung *Anmeldung am SSH mit Schlüssel*)

```
ssh-copy-id user@serverYY.dane.onl
```

- nachdem die Schlüssel kopiert wurden kann die Passwort-Authentisierung in der Datei `/etc/ssh/sshd_config` ausgeschaltet werden

```
[...]
PasswordAuthentication no
[...]
```

- und den SSH-Dienst neu starten

```
systemctl restart sshd
```

- Nun sollte auf dem Server nur noch eine Anmeldung per Schlüssel möglich sein

- Stelle sicher, das der SSH-Agent für die aktuelle Shell-Sitzung aktiv ist und lade die eigenen Schlüssel in den SSH-Agent

```
ssh-add -l
```

- Teste die Anmeldung an den Trainer-Server

```
ssh serverXX.dane.onl
```

- Teste, das eine Anmeldung mit dem eigenen Benutzer (`user`) von Server des Trainers auf den Server des Übungs-Partners (`serverYY`) nicht möglich ist

```
trainer-server$ ssh user@serverYY.dane.onl
```

- den Server des Trainers wieder verlassen

```
trainer-server$ exit
```

- Wir nehmen an, dass eine direkte Netzwerkverbindung zwischen dem eigenen Server und dem Server des benachbarten Teilnehmers **nicht** möglich ist, ein Login auf diesem Server muss über den Umweg des Trainer-Server geschehen (Jump-Host)
- Teste die SSH-Agent-Forwarding Funktion mit einer Anmeldung an den Jump-Host (Trainer-Server) und von dort aus einer Anmeldung an den Server des benachbarten Teilnehmers. Die zweite Anmeldung solle bei aktiven SSH-Agent ohne Passwortabfrage funktionieren

```
serverXX$ ssh -A user@serverXX.dane.onl # Trainer server
trainer-server$ ssh user@serverYY.dane.onl
```

3.16.8. SSH Agent-Forwarding vs. ProxyCommand

- Sicherheits-Probleme mit Agent-Forwarding!
  - https://heipei.github.io/2015/02/26/SSH-Agent-Forwarding-considered-harmful/ [https://heipei.github.io/2015/02/26/SSH-Agent-Forwarding-considered-harmful/]
  - https://michael.stapelberg.de/Artikel/ssh-conditional-tunneling [https://michael.stapelberg.de/Artikel/ssh-conditional-tunneling]
- Agent forwarding (`ssh -A user@serverYY.dane.onl`) kann von einem Angreifer auf dem "Jumphost" abgefangen werden (Socket zum Agent im /tmp-Verzeichnis des Jumphost).

AgentForwarding Alternative `ProxyCommand`

- vor OpenSSH 7.3

```
Host jumphost
    User nutzer
    Hostname 192.0.2.10 # Trainer Server
    Port 22

Host internalserver
    User user
    Hostname serverYY.dane.onl # Server des anderen Teilnehmers
    Port 22
```

```
    ProxyCommand ssh -q -W %h:%p jumphost
```

- seit OpenSSH 7.3

```
Host jumphost
    User nutzer
    Hostname 192.0.2.10 # Trainer Server
    Port 22

Host internalserver
    User user
    Hostname serverYY.dane.onl # Server des anderen Teilnehmers
    Port 22
    ProxyJump jumphost
```

- ProxyJump auf der Kommandzeile (seit OpenSSH 7.3)

```
ssh -J jumphost benutzer@internalserver
```

Übung:

- Erstelle eine lokale Konfigurationsdatei `\~/.ssh/config` mit der ProxyCommand Konfiguration (siehe oben)
- SSH-Verbindung durch den JumpHost (Trainer-Server) zum Server eines anderen Teilnehmers testen

```
ssh serverYY
```

3.16.9. SSH Tunnel

- Über SSH Tunnel können beliebige Ports über die SSH Verbindung geleitet werden
- Praktische Illustration zu local und remote tunnels: [ssh tunnels] (Quelle: https://iximiuz.com/ssh-tunnels/ssh-tunnels.png)
- Tunnel von Port 80 auf dem Server (serverYY) zu Port 8080 auf dem Client über serverXX

```
ssh -L 8080:serverYY:80 user@serverXX
ssh -L 8080:datenbank:80 user@serverXX
```

- Rückwärts-Tunnel: vom Ziel-System zurück auf den Client. Dieses Beispiel öffnet einen Tunnel von `localhost:8000` auf dem Zielsystem auf Port 8000 des SSH-Clients

```
ssh -R 8000:serverXX:8000 user@serverXX
```

- Socks-Proxy Forwarding (Web-Browsen durch den Tunnel), im Browser die Socks-Proxy Konfiguration auf 127.0.0.1 Port 8080 setzen

```
ssh -CD 127.0.0.1:8080 nutzer@server
```

Übung: SSH Tunnel

- Erstelle einen SSH-Tunnel von Port 8080 (lokal) auf die IPv4-Loopback-Adresse (127.0.0.XX; x=Teilnehmernummer, z.B. 42) des Trainer-Servers, Port 8000:

```
ssh -L 8080:127.0.0.XX:8000 user@serverXX.dane.onl # Server des Trainers
```

- Auf dem Trainer-Server erstellen wir eine minimale HTML-Datei

```
mkdir ~/webseiteXX
cd ~/webseiteXX
~/webseiteXX% echo "Dies ist der Webserver von NutzerXX" > index.html
```

- Auf dem Trainer-Server den Python3 Webserver auf der Loopback-Adresse, Port 8000, starten

```
~/webseiteXX% python3 -m http.server --bind 127.0.0.XX
```

- Starte den `links` Browser auf dem eigenen Server (ggf. ein weiteres Terminal öffnen), verbinde Dich mit der eigenen Loopback-Adresse Port 8080 (URL `links http://127.0.0.1:8080/`). Es sollte die minimale Webseite aus dem Heimverzeichnis auf dem Trainer-Server erscheinen

3.16.10. SSH RSA-SHA1 Schlüssel sind abgekündigt

- OpenSSH hat in früherern Versionen standardmässig Schlüssel mit dem RSA-SHA1 Algorithmus erzeugt. Dieser Algorithmus gilt inzwischen als unsicher und werden bald mit einer neuen Version von OpenSSH nicht mehr unterstützt
  - Daher sollten Benutzer von OpenSSH testen, ob sie noch RSA-SHA1 Schlüssel benutzen und diese ggf. durch neue Schlüssel austauschen
  - Test mit RSA-SHA1 abgeschaltet. Klappt die Anmeldung, so wird ein anderer Schlüssel benutzt. Schlägt die Anmeldung fehlt, so sollte ein neuer Schlüssel für diesen Benutzer erstellt werden:

    ```
    ssh -oHostKeyAlgorithms=-ssh-rsa user@host
    ```

  - Gibt es die folgende Fehlermeldung, dann benutzt der Server noch alte RSA-SHA1 Schlüssel und diese Schlüssel sind in der lokalen `known-hosts` Datei eingetragen:

```
    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    @    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
    IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
    Someone could be eavesdropping on you right now (man-in-the-middle attack)!
    It is also possible that a host key has just been changed.
    The fingerprint for the ECDSA key sent by the remote host is
    SHA256:/7MPZYXQhPQ0uwZaupxxUC0EDU7I2D+s8OQCLtN69rE.
    Please contact your system administrator.
    Add correct host key in /home/nutzer/.ssh/known_hosts to get rid of this message.
    Offending RSA key in /home/nutzer/.ssh/known_hosts:157
    ECDSA host key for ssh-host.example.com has changed and you have requested strict
checking.
    Host key verification failed.
```

- Der Server-Betreiber sollte den alten RSA-SHA1 Schlüssel vom Server entfernen
- Der Benutzer sollte den Hash-Eintrag des RSA-SHA1 aus der Datei `known-hosts` entfernen und dann den Hash des neuen Schlüssels mit dem Betreiber des Servers (oder via DNS, siehe unten) abgleichen
    - Empfohlene SSH Schlüssel-Algorithmen
- RFC 8332 – RSA SHA-2 signature algorithms rsa-sha2-256/512. [https://www.rfc-editor.org/rfc/rfc8332.txt]
- RFC 5656 – Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer [https://datatracker.ietf.org/doc/html/rfc5656] (ecdsa-sha2-nistp256/384/521)
- RFC 8709 – Ed25519 and Ed448 Public Key Algorithms for the Secure Shell (SSH) Protocol [https://datatracker.ietf.org/doc/html/rfc8709]
    - Informationen: OpenSSH 8.3 released (and ssh-rsa deprecation notice) [https://lwn.net/Articles/821544/]

3.16.11. Lokale SSH-Konfigurationsdatei

- Bei komplexen SSH-Konfigurationen ist es hilfreich, die SSH-Parameter pro Ziel-System in der lokalen SSH-Konfigurationsdatei für den Benutzer abzulegen. Fast alle Kommandozeilen-Optionen des SSH-Clients können in dieser Konfigurationsdatei festgelegt werden (siehe `man ssh`).
- Beispiel einer lokalen SSH-Konfigurationsdatei für einen Benutzer. Datei `\~/.ssh/config`

```
Host zielhost                                       # symbolischer Name
        Hostname zielhost.example.com               # echter DNS-Name
    User user                                     # Standard Benutzer
    Compression yes                               # gzip Kompression an
    VerifyHostKeyDNS yes                          # Host-Key per DNS prüfen
    Port 65123                                    # SSH läuft auf diesem Port
    ProxyJump jumphost.example.com                # Verbindung über Jumphost
```

Übung: lokale SSH-Client-Konfigurationsdatei

- Zeit: 15 Minuten
- Erstelle eine lokale SSH-Konfigurationsdatei `\~/.ssh/config` für die Verbindung auf die IP-Adresse des VM Servers
    - Vergebe den symbolischen Namen `vmXX` für den Server (XX= Teilnehmernummer)
    - SSH-Server läuft auf Port 22
    - Benutzername `user`, Passwort `linux-sicherheit`
    - Benutze die IP-Adresse des Servers als Hostnamen in der Konfiguraionsdatei (finde die IPv4-Adresse des Servers mit `dig` oder `nslookup`)

```
% dig serverXX.dane.onl a
```

- Die SSH Konfigurationsdatei sollte nur für den Benutzer lesbar sein (ggf. müssen die Dateiberechtigungen angepasst werden)

- teste die Verbindung per symbolischen Namen

```
ssh vmXX
```

- Beispiellösung

```
Host vmXX
  Hostname  192.0.2.10
  User user
  Port 22
```

Weitere Informationen und Tipps: Simplify Your Life With an SSH Config File

https://dev.to/paulmicheli/simplify-your-life-with-an-ssh-config-file-411g [https://dev.to/paulmicheli/simplify-your-life-with-an-ssh-config-file-411g]

3.16.12. VisualHostKey

- Der Konfigurationsparameter `VisualHostKey` in der Datei `/.ssh/config` zeigt eine visuelle Darstellung des Schlüssel-Hash jedes Ziel- und Jump-Host Servers an

```
VisualHostKey yes
```

3.16.13. SSH-PAM-Module

- `pam_ssh.so` — per SSH Key Passphrase anmelden und Schlüssel in den Agent laden
- `pam_sshauth.so` — lokale Anmeldung durch erfolgreiche SSH-Anmeldung an einen fernen SSH Server --> E-Mail mit Beispielkonfiguration

3.16.14. Übung: PAM-Anmeldung per SSH Passphrase

--> E-Mail wenn Loesung fuer ssh_pam auf dem Server

- installiere `pam_ssh`

```
apt install libpam-ssh
```

- prüfe, ob sich schon SSH-Schlüssel unter `/home/user/.ssh/` befinden. Wenn nicht, einen SSH-Schlüssel erstellen per `ssh-keygen` und für den Schlüssel eine Passphrase vergeben. Diese Passphrase sollte **nicht** das normale Benutzerpasswort sein.
- Konfiguriere `pam_ssh` als `sufficient` PAM-Modul für das grafische Login (Datei `/etc/pam.d/gdm-password =, Facility =auth`)
- Konfiguriere `pam_ssh` als `optional` PAM-Modul in `/etc/pam.d/common-session` (Facility `session`)
- Die grafische GUI verlassen und an der GUI eine Anmeldung per SSH-Passphrase versuchen
- Prüfen das der SSH-Agent gestartet ist und den/die Schlüssel geladen hat

```
ssh-add -l
```

3.16.15. Socks-SSH-Proxy

- Socks-Proxy Forwarding (Web-Browsen durch den Tunnel), im Browser die Socks-Proxy Konfiguration auf 127.0.0.1 Port 8080 setzen
  - Option `-C` schaltet Kompression der Daten im Tunnel ein
  - Option `-D` schaltet den Socks-Proxy für den SSH-Tunnel ein

```
ssh -C -D 127.0.0.1:8080 nutzer@server
```

3.16.16. SSH-Escape

- A tilde (\~) after an Enter is the escape sequence of SSH
- `~\~` Tilde senden
- `\~.` SSH-Verbindung sofort stoppen
- `\~<Ctrl-Z>` SSH Sitzung in den Hintergrund senden (mit fg wieder in den Vordergrund holen)
- `\~&` SSH als 'daemon' in den Hintergrund senden und vom Terminal ablösen
- `\~?` Hilfe anzeigen.
- `\~R` neuen Sitzungschlüssel aushandeln
- `\~C` SSH Komandozeile anzeigen (kann benutzt werden um neue Tunnel in schon bestehende Verbindungen einzubauen)
- `\~#` bestehende SSH-Tunnel anzeigen

3.16.17. Ausführbare Befehle über die `authorized_keys` beschränken

```
command="foobar" ssh-rsa AAAAB3Nza… calls "foobar" and only "foobar" upon every login
with this key
from="computer" ssh-rsa AAAAB3Nza… allows access with this key only from the host
"computer".
```

3.16.18. Verwaltung von SSH authorized_keys auf dem Server

- Authorized-Keys via FUSE und DNS http://jpmens.net/2012/12/18/ssh-public-keys-on-a-dns-fuse/ [http://jpmens.net/2012/12/18/ssh-public-keys-on-a-dns-fuse/]
- Authorized-Keys per TXT-Record im DNS (ohne FUSE) https://blog.cloudflare.com/flexible-secure-ssh-with-dnssec/ [https://blog.cloudflare.com/flexible-secure-ssh-with-dnssec/]

3.16.19. Ein CHROOT für SSH

- Eine Gruppe für die CHROOT SSH Benutzer erstellen

```
groupadd sshusers
useradd -m -s /bin/bash -g sshusers sshguy
passwd sshguy
```

- Wir erzeugen ein neues **root** Verzeichnis, das chroot-Verzeichnis

```
mkdir -p /var/ssh-chroot/{dev,etc,lib,lib64,usr,bin,home}
mkdir -p /var/ssh-chroot/usr/bin
chown root: /var/ssh-chroot
```

- Die Gerätedatei `/dev/null` anlegen

```
mknod -m 666 /var/ssh-chroot/dev/null c 1 3
```

- Das Heimverzeichnis des Benutzers in das **chroot** kopieren

```
cp -R /home/sshguy /var/ssh-chroot/home/
chown -R sshguy /var/ssh-chroot/home/
```

- Wir kopieren einige Dateien aus `/etc` in das Verzeichnis `etc` innerhalb des `chroot`

```
cd /var/ssh-chroot/etc
cp /etc/ld.so.cache /etc/ld.so.conf /etc/nsswitch.conf /etc/resolv.conf .
```

- Der Benutzer soll `bash` und `ls` ausführen können, also kommen diese Dateien in die **chroot**

```
cd ../bin
cp /bin/ls /bin/bash .
```

- Für jedes Programm im **chroot** müssen wir nun die dynamischen Bibliotheken in das **chroot** Verzeichnis kopieren. Der `ldd` Befehl listet alle Bibliotheken

```
ldd /bin/ls
ldd /bin/bash
```

- Wir haben ein Skript, welches die Bibliotheken für ein Programm in das **chroot** kopiert:

```
mkdir ~/bin
$EDITOR ~/bin/l2chroot
----
#!/bin/bash
# Use this script to copy shared (libs) files to Apache/Lighttpd chrooted
# jail server.
# ------------------------------------------------------------------------
# Written by nixCraft <http://www.cyberciti.biz/tips/>
# (c) 2006 nixCraft under GNU GPL v2.0+
# + Added ld-linux support
# + Added error checking support
# ------------------------------------------------------------------------
# See url for usage:
# http://www.cyberciti.biz/tips/howto-setup-lighttpd-php-mysql-chrooted-jail.html
# ------------------------------------------------------------------------
# Set CHROOT directory name
BASE="/var/ssh-chroot"

[ ! -d $BASE ] && mkdir -p $BASE || :

# iggy ld-linux* file as it is not shared one
FILES="$(ldd $1 | awk '{ print $3 }' |egrep -v ^'\(')"
```

```
echo "Copying shared files/libs to $BASE..."
for i in $FILES
do
  d="$(dirname $i)"
  [ ! -d $BASE$d ] && mkdir -p $BASE$d || :
  /bin/cp $i $BASE$d
done

# copy /lib/ld-linux* or /lib64/ld-linux* to $BASE/$sldlsubdir
# get ld-linux full file location
sldl="$(ldd $1 | grep 'ld-linux' | awk '{ print $1}')"
# now get sub-dir
sldlsubdir="$(dirname $sldl)"

if [ ! -f $BASE$sldl ];
then
  echo "Copying $sldl $BASE$sldlsubdir..."
  /bin/cp $sldl $BASE$sldlsubdir
else
  :
fi
```

- Script ausführbar machen

```
chmod +x ~/bin/l2chroot
```

- Das Script benutzen, um die Bibliotheken in das **chroot** zu kopieren

```
l2chroot /var/ssh-chroot/bin/ls
l2chroot /var/ssh-chroot/bin/bash
```

- Prüfen, das Bibliotheks-Dateien kopiert wurden

```
ls -l /var/ssh-chroot/lib64
ls -l /var/ssh-chroot/lib/x86_64-linux-gnu/
```

- Einen Hardlink für BASH in `/usr/bin` im chroot erstellen

```
ln bin/bash usr/bin/
```

- Manueller Test der **chroot** Umgebung:

```
chroot /var/ssh-chroot/ /bin/bash
```

- Chroot für die Gruppe **sshusers** in der SSH-Konfiguration einrichten (Konfiguration am Ende der Datei anfügen)

```
$EDITOR /etc/ssh/sshd_config
----
Match group sshusers
          ChrootDirectory /var/ssh-chroot/
          X11Forwarding no
          AllowTcpForwarding no
      PasswordAuthentication yes
```

```
----
```

- SSH-Daemon neu starten

```
systemctl restart sshd
```

- Ausprobieren:

```
ssh sshguy@localhost
```

- Fertig!

3.16.20. Daten mit SSH-Schlüsseln signieren

SSHIGN

- https://blog.mnus.de/2019/11/announcing-sshign/ [https://blog.mnus.de/2019/11/announcing-sshign/]

3.16.21. Daten mit SSH-Schlüsseln verschlüsseln

- Encrypt and decrypt a file using SSH keys https://www.bjornjohansen.com/encrypt-file-using-ssh-key [https://www.bjornjohansen.com/encrypt-file-using-ssh-key]

SSH-vault

- https://ssh-vault.com/ [https://ssh-vault.com/]

JASS

- https://github.com/jschauma/jass [https://github.com/jschauma/jass]

3.16.22. TPM chip protecting SSH keys – properly

https://blog.habets.se/2013/11/TPM-chip-protecting-SSH-keys---properly.html [https://blog.habets.se/2013/11/TPM-chip-protecting-SSH-keys---properly.html] https://github.com/ThomasHabets/simple-tpm-pk11 [https://github.com/ThomasHabets/simple-tpm-pk11]

3.16.23. Alerting or notifying on SSH logins / PAM

- Original von Jan-Piet Mens mit Benachrichtigung via MQTT: https://jpmens.net/2018/03/25/alerting-on-ssh-logins/ [https://jpmens.net/2018/03/25/alerting-on-ssh-logins/]
- *zentralisierte* Benachrichtigung bei SSH-Logins auf einem Rechner der eigenen IT-Infrastruktur. Kommunikation zwischen SSH-Server und zentraler Monitoring-Instatnz über leichtgewichtiges UDP, E-Mail Kommunikationsparameter werden zentral verwaltet.
- Quellcode des `hare` Programms (PAM-Modul für die SSH-Server)

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

```
#include <netdb.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <memory.h>
#include <sys/socket.h>

/*
 * hare.c (C)2018 by Jan-Piet Mens <jp@mens.de>
 */

#define PORTNO  8035

#define env(K)  ( getenv(K) ? getenv(K) : "<unknown>" )

int main(int argc, char **argv)
{
        struct sockaddr_in servaddr;
        unsigned long addr;
        int sockfd;
        char *host = argv[1], buf[BUFSIZ], myhostname[BUFSIZ];
        char *pamtype = env("PAM_TYPE");        /* Linux */
        char *pamsm = env("PAM_SM_FUNC");       /* FreeBSD
https://www.freebsd.org/cgi/man.cgi?query=pam_exec */

        if (strcmp(pamtype, "open_session") != 0 && strcmp(pamsm,
"pam_sm_open_session") != 0) {
                fprintf(stderr, "Neither PAM open_session nor pam_sm_open_session
detected\n");
                return 0;
        }

        if (argc != 2) {
                fprintf(stderr, "Usage: %s address\n", *argv);
                exit(2);
        }

        if (gethostname(myhostname, sizeof(myhostname)) != 0)
                strcpy(myhostname, "?");

        snprintf(buf, sizeof(buf), "%s login to %s from %s via %s",
                env("PAM_USER"),
                myhostname,
                env("PAM_RHOST"),
                env("PAM_SERVICE"));

        addr = inet_addr(host);

        memset(&servaddr, 0, sizeof(servaddr));
        servaddr.sin_family = AF_INET;
        servaddr.sin_port = htons(PORTNO);
        memcpy((void *)&servaddr.sin_addr, (void *)&addr, sizeof(addr));

        sockfd = socket(AF_INET, SOCK_DGRAM, 0);
```

```
        if (sendto(sockfd, (void *)buf, strlen(buf), 0, (struct sockaddr *) &servaddr,
sizeof(servaddr)) < 0) {
                perror("sendto");
        }
        return (0);
}
```

- Übersetzen und Installieren des `hare` Programms (ggf. das Paket `gcc` und `build-essential` installieren)

```
gcc -O2 -o hare hare.c
sudo cp hare /usr/local/sbin
```

- das `hare` Programm in die PAM-Konfiguration `/etc/pam.d/sshd` eintragen. Ziel-IP ist der Server des Trainers (192.168.1.241)

```
[...]
session    optional     pam_exec.so /usr/local/sbin/hare <ip-des-hare-servers>
[...]
```

- der zentrale Python-Server `/usr/local/bin/hared.py`, mit der Funktion pro SSH-Login eine Benachrichtigungs-Mail zu versenden (läuft auf dem Server des Trainers)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import socket
import json
import smtplib, ssl
import random
import string
import datetime

__author__    = 'Jan-Piet Mens <jp()mens.de> / Carsten Strotmann
<carsten()strotmann.de>'

myhostname     = "debian.example.com" # Domain-Name des E-Mail Senders (diese Maschine)
smtp_server    = "mail.example.com" # Name of the Mail-Server
smtp_port      = "587"  # SMTP with STARTTLS
sender_email   = "servernotification@example.com"
receiver_email = "admin@example.com"
smtp_user      = "admin"
smtp_password  = "secret"

server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind(('', 8035))

while True:
    message, address = server_socket.recvfrom(1024)
    host, port = address

    data = {
        'host' : host,
        'msg'  : message,
```

```
    }

    js = json.dumps(str(data))
    # print(js)


    message = "To: <{}>\n".format(receiver_email)
    message = message + "From: <{}>\n".format(sender_email)
    message = message + "Message-Id:
<{}@example.com>\n".format(''.join(random.choices(string.ascii_uppercase +
string.digits, k=10)))
    time = datetime.datetime.now()
    message = message + "Date: {} \n".format(time.strftime('%d %b %Y  %H:%M:%S %z'))
    message = message + "Subject: SSH Login to {}\n\n".format(data["host"])
    message = message + "SSH Login into {} detected, Message: {}".format(data["host"],
data["msg"])

    # Create a secure SSL context
    context = ssl.create_default_context()
    # if the TLS certificate of the mail server does not match
    # the mail servers hostname or domain-name, disable hostname checking
    # context.check_hostname = False
    server = smtplib.SMTP()
    # Try to log in to server and send email
    try:
        server.connect(smtp_server,smtp_port)      # Connect to the server
        server.ehlo(myhostname)                    # Send EHLO
        server.starttls(context=context)           # TLS Secure the connection
        server.ehlo(myhostname)                    # Another EHLO, now TLS secured
        server.login(smtp_user, smtp_password)     # Login to the server

        server.sendmail(sender_email, receiver_email, message) # send the mail
    except Exception as e:
        # Print any error messages to stdout
        print("Error: ", e)
    finally:
        if (server):
            server.quit()
```

3.16.24. Vorschläge für eine sichere SSH-Server Konfiguration

- Quellen
  - SSH Security https://stribika.github.io/2015/01/04/secure-secure-shell.html [https://stribika.github.io/2015/01/04/secure-secure-shell.html]
  - Secure Secure Shell Wiki: https://github.com/stribika/stribika.github.io/wiki/Secure-Secure-Shell [https://github.com/stribika/stribika.github.io/wiki/Secure-Secure-Shell]
- SSH Server Konfiguration öffnen

```
$EDITOR /etc/ssh/sshd_config
```

- Nur Protokoll Version 2 erlauben

```
Protocol 2
```

- Sichere Schlüssel–Austausch–Verfahren

```
KexAlgorithms curve25519-sha256@libssh.org,diffie-hellman-group-exchange-sha256
```

- neue DH Moduli für "diffie–hellman–group–exchange–sha256" (Achtung, dauert lange!)
  Hintergründe unter http://weakdh.org [http://weakdh.org]

```
ssh-keygen -G /etc/ssh/moduli.all -b 4096
ssh-keygen -T /etc/ssh/moduli.safe -f /etc/ssh/moduli.all
mv /etc/ssh/moduli.safe /etc/ssh/moduli
rm /etc/ssh/moduli.all
# Bei SELinux Systemen
chcon -v --type=etc_t /etc/ssh/moduli
```

- EC–ED25519–Kurve Schlüssel bevorzugen

```
HostKey /etc/ssh/ssh_host_ed25519_key
HostKey /etc/ssh/ssh_host_rsa_key
```

- Sichere symmetrische Verschlüsselungs–Algorithmen

```
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-
gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
```

- Sichere Message–Authenication–Codes (MAC)

```
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-
etm@openssh.com,hmac-sha2-512,hmac-sha2-256,umac-128@openssh.com
```

- Host–Keys neu erzeugen (Achtung: Clients werden Warnungen bekommen, Kunden und Kollegen informieren!)

```
cd /etc/ssh
rm ssh_host_*key*
ssh-keygen -t ed25519 -f ssh_host_ed25519_key < /dev/null
# rsa-sha2-512
ssh-keygen -t rsa -b 4096 -f ssh_host_rsa_key < /dev/null
```

- Nach dem Erstellen neuer Schlüssel den SSH Server neu starten

```
systemctl restart sshd
```

- SSH Client Parameter testen

```
ssh -Q cipher
ssh -Q cipher-auth
ssh -Q mac
ssh -Q kex
ssh -Q key
```

- Jetzt nochmal per SSH auf dem fernen Rechner einloggen, `-v` Zeigt die Kommunikationsparameter an

```
user$ ssh -v nutzer@serverYY.dane.onl
```

- Hier sollte jetzt eine Warnung ausgegeben worden sein, da die Host-Schlüssel nicht mehr mit den Fingerprints in der `known_hosts` Datei übereinstimmen. Den Admin des fernen Servers fragen, ob er die Schlüssel auf dem Rechner neu erstellt hat. Wenn der Admin dies bestätigt, dann er Texteditor den alten Fingerprint aus der `\~/.ssh/known_hosts` Datei löschen.
- SSH Client `\~/.ssh/config`

```
VerifyHostKeyDNS yes
VisualHostKey yes
```

3.16.25. Links zum Thema SSH

- (Basic) SSH Keys Part 1: Simple setup + ProxyJump http://bad.network/basic-ssh-keys-part-1-simple-setup-proxyjump.html [http://bad.network/basic-ssh-keys-part-1-simple-setup-proxyjump.html]
- (Basic) SSH Keys Part 2: Agent forwarding, Adding Keys http://bad.network/basic-ssh-keys-part-2-agent-forwarding-adding-keys.html [http://bad.network/basic-ssh-keys-part-2-agent-forwarding-adding-keys.html]
- Using a U2F/FIDO key with OpenSSH https://xosc.org/u2fandssh.html [https://xosc.org/u2fandssh.html]
- Cryptographic Signing using ssh-keygen(1) with a FIDO Authenticator https://undeadly.org/cgi?action=article;sid=20201016053038 [https://undeadly.org/cgi?action=article;sid=20201016053038]
- SCP – Familiar, Simple, Insecure, and Slow https://goteleport.com/blog/scp-familiar-simple-insecure-slow/ [https://goteleport.com/blog/scp-familiar-simple-insecure-slow/]
- OpenSSH 9.0 Release Notes

> This release switches scp(1) from using the legacy scp/rcp protocol to using the SFTP protocol by default.

- https://www.openssh.com/txt/release-9.0 [https://www.openssh.com/txt/release-9.0]

3.17. SKRIPT SICHERHEIT

- Skripte, wenn möglich, nicht als Benutzer `root` laufen lassen.
  - Grössere Skripte aufteilen, Funktionen welche Root-Rechte benötigen auslagern.
  - Root-Funktionen ggf. auf Systemdienste auslagern.
  - Aktionen über System-Kanäle delegieren (DBus, Systemd, Polkit).
  - Skripte nicht per `SUID`-Rechten als `root` laufen lassen.
- Prüfung von Eingabedaten:
  - Die Prüfung von externen Daten auf Angriffe ist schwer, Autoren von Skripten sollte immer davon ausgehen das die Validierung der Eingabedaten nicht 100% abgesichert werden kann und die Skripte entsprechend in isolierter Umgebung ausführen.
  - Kein *EVAL* benutzen, d.h. keine Daten als Programmcode ausführen

    - Eingabedaten von externen Werkzeugen parsen lassen, nicht im Skript einen Parser schreiben.
- Sub–Programme und –Shells in Skripten immer mit vollem Programmpfad referenzieren. Dies verhindert das Angreifer gleichnamige, bösartige Programme im Suchpfad einschleusen.
- Sktipte mittels Systemd absichern (`systemd-run` oder als Systemd–Unit starten), nicht verwendete Rechte entziehen (CGroups, Capabilities).
- Skripte in eigenen Namespaces/Containern starten.
    - Zugriff auf das Dateisystem einschränken.
    - Zugriff auf das Netzwerk einschränken.
    - Zugriff auf Prozesse einschränken.
- Zugriff auf Skripte und Daten per Audit–Subsystem absichern (eigene Audit–Regeln).
- Skripte oder Sub–Skripte unter Kontrolle von `sudo` ausführen.
    - `sudo` Erzeugt Audit–Daten im Audit–Log.
- Skripte mittels Hashes gegen Manipulation schützen.
    - Hashes via `sudo` vor der Ausführung prüfen lassen.

3.17.1. Übung: Shell Skripte mit ShellCheck prüfen

- ShellCheck ist ein Programm welches Sicherheits–Probleme (und andere Probleme) in Shell–Skripten meldet

```
apt install shellcheck
```

- Ein Shell–Skript laden

```
wget https://raw.githubusercontent.com/menandmice-services/dns-resolver-
monitoring/refs/heads/master/dnsresolver-check.sh
```

- Dieses Shell–Skript prüfen

```
shellcheck dnsresolver-check.sh
```

- Ausgabe

```
In dnsresolver-check.sh line 58:
    echo -n " $@ "
             ^-- SC2145 (error): Argument mixes string and array. Use * or separate
argument.


In dnsresolver-check.sh line 86:
            local FILE=$(readlink -m "${BASH_SOURCE[1]}")
                  ^--^ SC2155 (warning): Declare and assign separately to avoid masking
return values.


In dnsresolver-check.sh line 98:
```

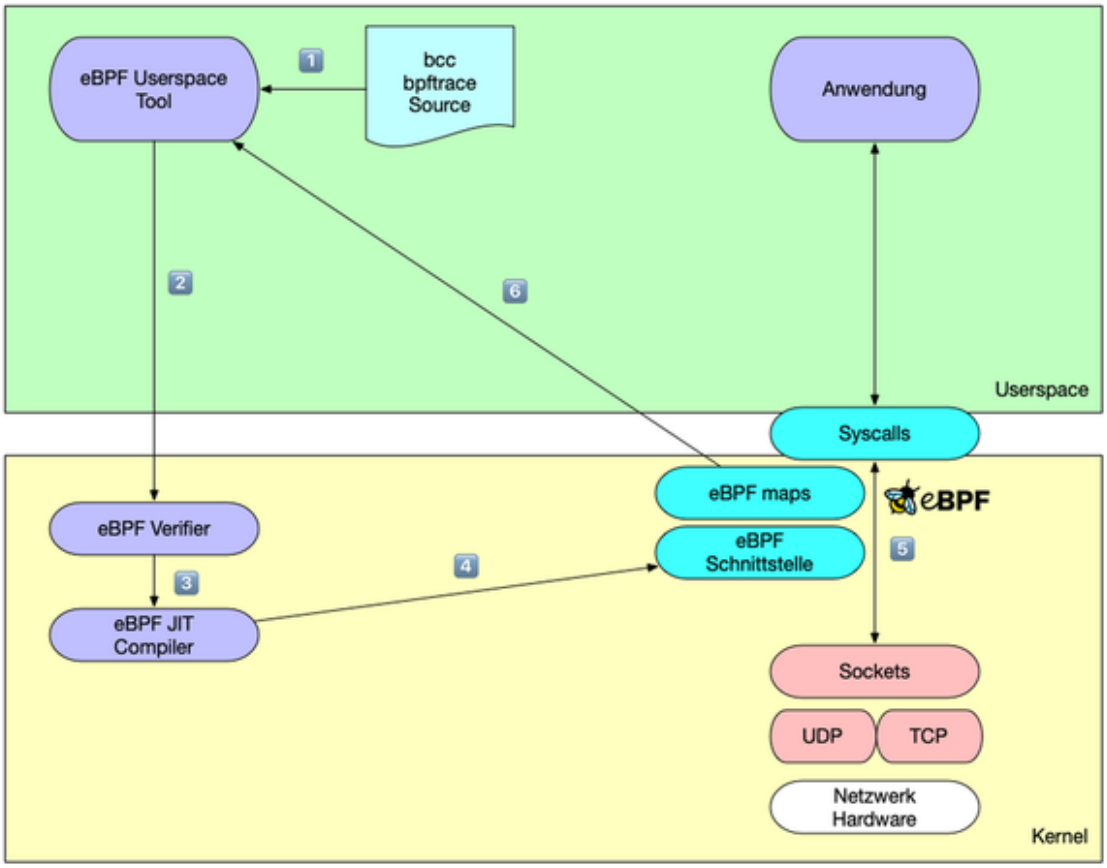
```
    [[ $STEP_OK -eq 0 ]]  && echo_success || echo_failure
                          ^-- SC2015 (info): Note that A && B || C is not if-then-else.
C may run when A is true.
[...]
For more information:
  https://www.shellcheck.net/wiki/SC2145 -- Argument mixes string and array. ...
  https://www.shellcheck.net/wiki/SC2155 -- Declare and assign separately to ...
  https://www.shellcheck.net/wiki/SC2015 -- Note that A && B || C is not if-t...
```

3.17.2. Links

- https://moldstud.com/articles/p-shell-scripting-security-how-to-protect-your-scripts-from-vulnerabilities [https://moldstud.com/articles/p-shell-scripting-security-how-to-protect-your-scripts-from-vulnerabilities]
- https://developer.apple.com/library/archive/documentation/OpenSource/Conceptual/ShellScripting/ShellScriptSecurity/ShellScriptSecurity.html [https://developer.apple.com/library/archive/documentation/OpenSource/Conceptual/ShellScripting/ShellScriptSecurity/ShellScriptSecurity.html]
- https://github.com/koalaman/shellcheck#user-content-installing [https://github.com/koalaman/shellcheck#user-content-installing]

3.18. EBPF/BCC

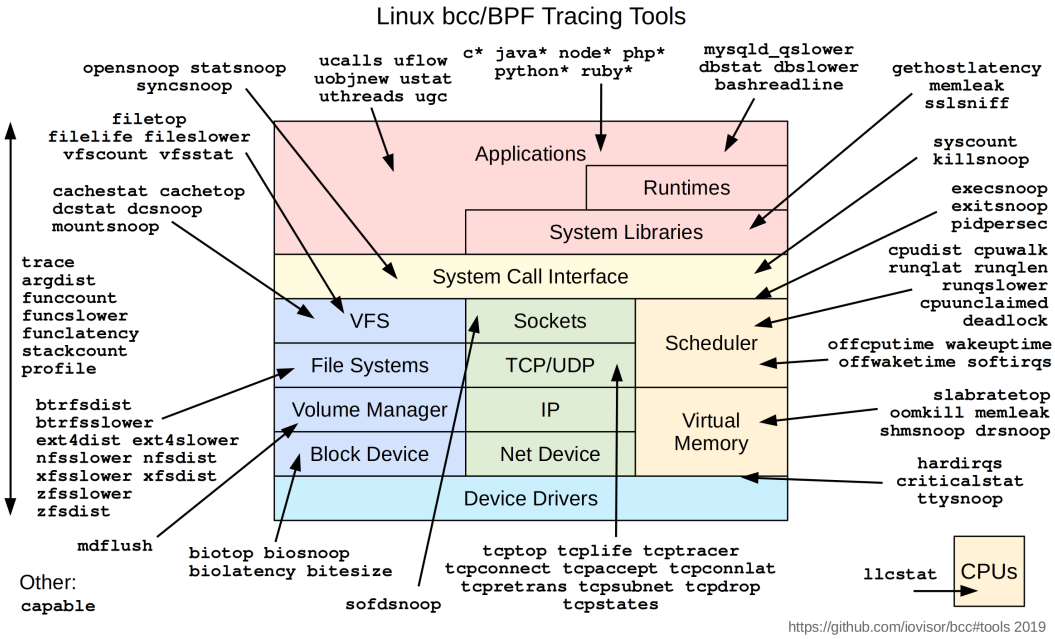
- eBPF ist die *extended Berkeley Packet Filter* Infrastruktur im Linux Kernel
- eBPF ist eine Weiterentwicklung der Berkeley Packet Filter Technologie (*classic* BPF = cBPF) https://en.wikipedia.org/wiki/Berkeley_Packet_Filter [https://en.wikipedia.org/wiki/Berkeley_Packet_Filter]
- eBPF ist eine Technologie im Linux-Kernel, welche Sandbox-Programme in einem Betriebssystem-Kernel ausführen kann.
    - eBPF wird verwendet, um die Fähigkeiten des Kernels sicher und effizient zu erweitern, ohne dass der Kernel-Quellcode geändert oder Kernel-Module geladen werden müssen.
    - eBPF kann neben Netzwerk-Paketen auch andere Daten aus dem Linux-Kernel überwachen und manipulieren



- Einsatzgebiete:
  - Netzwerksicherheit (erweiterte Firewall-Funktionen)
  - Host-Sicherheit
  - Forensik
  - Fehler-Diagnose
  - Performance-Messungen
- eBPF wird derzeit von Microsoft auf die Windows Betriebssysteme portiert
  - Making eBPF work on Windows https://cloudblogs.microsoft.com/opensource/2021/05/10/making-ebpf-work-on-windows/ [https://cloudblogs.microsoft.com/opensource/2021/05/10/making-ebpf-work-on-windows/]
  - eBPF implementation that runs on top of Windows https://github.com/microsoft/ebpf-for-windows [https://github.com/microsoft/ebpf-for-windows]

3.18.1. BCC Übersicht

- BCC ist die *BPF Compiler Collection*, eine Sammlung von Tools und eBPF Programmen



3.18.2. BCC Installieren (inkl. der Kernel–Header)

- BCC Programme sind abhängig von der jeweiligen Kernel–Version und müssen daher auf der Maschine mit Hilfe der aktuellen Kernel–Header kompiliert werden. Das Kompilieren geschied im Hintergrund beim Aufruf des Wrapper–Scripts.

```
# apt install bpfcc-tools bpfcc-lua bpftrace python3-bpfcc
# apt install linux-headers-$(uname -r)
```

3.18.3. BCC Tools für Linux–Server

- Unter Debian enden alle BCC Programme auf `-bpfcc`:

```
# ls -l /usr/sbin/*bpfcc
-rwxr-xr-x 1 root root 34536 Feb  4  2019 /usr/sbin/argdist-bpfcc
-rwxr-xr-x 1 root root  1648 Feb  4  2019 /usr/sbin/bashreadline-bpfcc
-rwxr-xr-x 1 root root  4231 Feb  4  2019 /usr/sbin/biolatency-bpfcc
-rwxr-xr-x 1 root root  5066 Feb  4  2019 /usr/sbin/biosnoop-bpfcc
-rwxr-xr-x 1 root root  6387 Feb  4  2019 /usr/sbin/biotop-bpfcc
-rwxr-xr-x 1 root root  1721 Feb  4  2019 /usr/sbin/bitesize-bpfcc
-rwxr-xr-x 1 root root  2453 Feb  4  2019 /usr/sbin/bpflist-bpfcc
-rwxr-xr-x 1 root root  6339 Feb  4  2019 /usr/sbin/btrfsdist-bpfcc
-rwxr-xr-x 1 root root 10101 Feb  4  2019 /usr/sbin/btrfsslower-bpfcc
-rwxr-xr-x 1 root root  4674 Feb  4  2019 /usr/sbin/cachestat-bpfcc
[...]
```

- Im folgenden werden einige BCC Tools vorgestellt, welche für die Sicherheit von Linux–Servern interessant sind

opensnoop

- Das BCC-Programm `opensnoop-bpfcc` zeigt systemweit an, welche Prozesse auf Dateien zugreifen (lesend und schreibend):

```
# opensnoop-bpfcc
PID    COMM               FD ERR PATH
16884  links               3   0 /lib/x86_64-linux-gnu/liblz.so.1
16884  links               3   0 /lib/x86_64-linux-gnu/liblzma.so.5
16884  links               3   0 /lib/x86_64-linux-gnu/libbz2.so.1.0
16884  links               3   0 /lib/x86_64-linux-gnu/libbrotlidec.so.1
16884  links               3   0 /lib/x86_64-linux-gnu/libz.so.1
16884  links               3   0 /lib/x86_64-linux-gnu/libssl.so.1.1
16884  links               3   0 /lib/x86_64-linux-gnu/libcrypto.so.1.1
16884  links               3   0 /lib/x86_64-linux-gnu/libgpm.so.2
16884  links               3   0 /lib/x86_64-linux-gnu/libc.so.6
16884  links               3   0 /lib/x86_64-linux-gnu/libm.so.6
16884  links               3   0 /lib/x86_64-linux-gnu/libbrotlicommon.so.1
16884  links               3   0 /lib/x86_64-linux-gnu/libdl.so.2
16884  links              -1   2 /etc/links.cfg
16884  links              -1   2 /etc/.links.cfg
16884  links              -1   2 /root/.links2/links.cfg
16884  links              -1   2 /root/.links2/.links.cfg
16884  links              -1   2 /root/.links2/html.cfg
16884  links              -1   2 /root/.links2/.html.cfg
16884  links              -1   2 /root/.links2/user.cfg
16884  links              -1   2 /root/.links2/.user.cfg
16884  links               8   0 /root/.links2/bookmarks.html
16884  links               8   0 /root/.links2/links.his
16884  links               8   0 /dev/pts/1
16884  links              -1   2 /dev/gpmctl
11348  tmux: server        7   0 /proc/16884/cmdline
11348  tmux: server        7   0 /proc/16880/cmdline
11348  tmux: server        7   0 /proc/16880/cmdline
```

statsnoop

- Das BCC-Programm `statsnoop-bpfcc` zeigt systemweit Dateizugriffe mit den `stat` Systemaufruf:

```
# statsnoop-bpfcc
PID    COMM               FD ERR PATH
11348  tmux: server        0   0 /etc/localtime
11349  bash                0   0 .
11349  bash               -1   2 /usr/local/sbin/stat
11349  bash               -1   2 /usr/local/bin/stat
11349  bash               -1   2 /usr/sbin/stat
11349  bash                0   0 /usr/bin/stat
11349  bash                0   0 /usr/bin/stat
11349  bash                0   0 /usr/bin/stat
11349  bash                0   0 /usr/bin/stat
11349  bash                0   0 /usr/bin/stat
11349  bash                0   0 /usr/bin/stat
16893  stat               -1   2 /sys/fs/selinux
```

```
16893  stat              -1   2 /selinux
```

memleak

- Das BCC-Programm `memleak-bpfcc` zeigt Memory-Leaks auf und kann benutzt werden, um Programme mit fehlerhafter Speicherverwaltung zu finden

```
[12:03:09] Top 10 stacks with outstanding allocations:
        40 bytes in 5 allocations from stack
                __kmalloc_node+0x171 [kernel]
        64 bytes in 1 allocations from stack
                __kmalloc_track_caller+0x163 [kernel]
        64 bytes in 1 allocations from stack
                kmem_cache_alloc_node+0x152 [kernel]
        320 bytes in 5 allocations from stack
                kmem_cache_alloc_node_trace+0x14e [kernel]
                __get_vm_area_node+0x7a [kernel]
        480 bytes in 5 allocations from stack
                kmem_cache_alloc_node_trace+0x14e [kernel]
                alloc_vmap_area+0x75 [kernel]
                alloc_vmap_area+0x75 [kernel]
                __get_vm_area_node+0xb0 [kernel]
                __vmalloc_node_range+0x67 [kernel]
                __vmalloc+0x30 [kernel]
                bpf_prog_alloc+0x40 [kernel]
                bpf_prog_load+0xf3 [kernel]
                __x64_sys_bpf+0x279 [kernel]
                do_syscall_64+0x53 [kernel]
                entry_SYSCALL_64_after_hwframe+0x44 [kernel]
        1632 bytes in 14 allocations from stack
                __kmalloc+0x16e [kernel]
        4840 bytes in 19 allocations from stack
                kmem_cache_alloc+0x149 [kernel]
        14592 bytes in 30 allocations from stack
                kmem_cache_alloc_trace+0x14c [kernel]
        21913600 bytes in 240 allocations from stack
                __alloc_pages_nodemask+0x1e0 [kernel]
+
```

mountsnoop

- Das BCC-Programm `mountsnoop-bpfcc` zeigt alle Aufrufe der `mount` und `umount` Systemaufrufe (Systemcalls):

```
# mountsnoop-bpfcc
COMM             PID     TID     MNT_NS      CALL
umount           16930   16930   4026531840  umount("/boot/efi", 0x0) = 0
mount            16932   16932   4026531840  mount("/dev/sda15", "/boot/efi", "vfat",
MS_NOSUID|MS_NOEXEC|MS_REMOUNT|MS_MANDLOCK|MS_DIRSYNC|MS_NOATIME|MS_NODIRATIME|MS_BIND|
MS_MOVE|MS_REC|MS_I_VERSION|MS_NOUSER|0x7f9a00000300, "") = 0
```

execsnoop

- Das BCC- Programm `execsnoop` protokolliert alle neu gestarteten Prozesse des Linux-Systems:

```
# execsnoop-bpfcc
PCOMM            PID    PPID   RET ARGS
systemd-getty-g  16948  16941    0 /usr/lib/systemd/system-generators/systemd-getty-
generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
systemd-gpt-aut  16949  16941    0 /usr/lib/systemd/system-generators/systemd-gpt-auto-
generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
cloud-init-gene  16942  16941    0 /usr/lib/systemd/system-generators/cloud-init-
generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
systemd-hiberna  16950  16941    0 /usr/lib/systemd/system-generators/systemd-
hibernate-resume-generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
lvmconfig        16951  16943    0
systemd-detect-  16952  16942    0 /usr/bin/systemd-detect-virt --container --quiet
systemd-system-  16955  16941    0 /usr/lib/systemd/system-generators/systemd-system-
update-generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
systemd-run-gen  16954  16941    0 /usr/lib/systemd/system-generators/systemd-run-
generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
systemd-sysv-ge  16956  16941    0 /usr/lib/systemd/system-generators/systemd-sysv-
generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
systemd-veritys  16957  16941    0 /usr/lib/systemd/system-generators/systemd-
veritysetup-generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
systemd-rc-loca  16953  16941    0 /usr/lib/systemd/system-generators/systemd-rc-local-
generator /run/systemd/generator /run/systemd/generator.early
/run/systemd/generator.late
systemd-detect-  16958  16942    0 /usr/bin/systemd-detect-virt --container --quiet
ds-identify      16959  16942    0 /usr/lib/cloud-init/ds-identify
mkdir            16960  16942    0 /usr/bin/mkdir -p
/run/systemd/generator.early/multi-user.target.wants
ln               16961  16942    0 /usr/bin/ln -snf /lib/systemd/system/cloud-
init.target /run/systemd/generator.early/multi-user.target.wants/cloud-init.target
```

killsnoop

- Das BCC-Programm `killsnoop-bpfcc` zeichnet alle Aufrufe des Systemaufruf `kill` auf:

```
# killsnoop-bpfcc
TIME      PID    COMM             SIG  TPID   RESULT
12:46:34  16998  pkill            1    632    0
```

pidpersec

- Das BCC-Programm `pidpersec-bpfcc` zeichnet die Anzahl der neuen Prozesse pro Sekunde

auf

```
# pidpersec-bpfcc
Tracing... Ctrl-C to end.
12:48:11: PIDs/sec: 0
12:48:12: PIDs/sec: 23
12:48:13: PIDs/sec: 3
12:48:14: PIDs/sec: 0
```

3.18.4. bpftrace

- `bpftrace` ist eine kleine Programmiersprache (ähnlich wie `awk`), welche es Administratoren ermöglicht, kleine eBPF Scripts zu schreiben
  - Homepage https://bpftrace.org/ [https://bpftrace.org/]
  - Github Source Repository https://github.com/iovisor/bpftrace [https://github.com/iovisor/bpftrace]
- Histogram-Daten werden in der Regel nach dem Beenden des Script mit `CTRL+C` ausgegeben
- Unter Debian liegen die mitgelieferten `bpftrace` Programme unterhalb von `/usr/sbin/` und haben die Dateiendung `.bt` (für bpftrace)

```
# ls /usr/sbin/*.bt
/usr/sbin/bashreadline.bt  /usr/sbin/execsnoop.bt       /usr/sbin/pidpersec.bt
/usr/sbin/tcpconnect.bt
/usr/sbin/biolatency.bt    /usr/sbin/gethostlatency.bt  /usr/sbin/runqlat.bt
/usr/sbin/tcpdrop.bt
/usr/sbin/biosnoop.bt      /usr/sbin/killsnoop.bt       /usr/sbin/runqlen.bt
/usr/sbin/tcpretrans.bt
/usr/sbin/bitesize.bt      /usr/sbin/loads.bt           /usr/sbin/statsnoop.bt
/usr/sbin/vfscount.bt
/usr/sbin/capable.bt       /usr/sbin/mdflush.bt         /usr/sbin/syncsnoop.bt
/usr/sbin/vfsstat.bt
/usr/sbin/cpuwalk.bt       /usr/sbin/oomkill.bt         /usr/sbin/syscount.bt
/usr/sbin/writeback.bt
/usr/sbin/dcsnoop.bt       /usr/sbin/opensnoop.bt       /usr/sbin/tcpaccept.bt
/usr/sbin/xfsdist.bt
```

- Beispiel eines `bpftrace` Scriptes

```
# cat /usr/sbin/syscount.bt
#!/usr/bin/env bpftrace
/*
 * syscount.bt  Count system callls.
 *              For Linux, uses bpftrace, eBPF.
 *
 * This is a bpftrace version of the bcc tool of the same name.
 * The bcc versions translates syscall IDs to their names, and this version
 * currently does not. Syscall IDs can be listed by "ausyscall --dump".
 *
 * Copyright 2018 Netflix, Inc.
 * Licensed under the Apache License, Version 2.0 (the "License")
 *
 * 13-Sep-2018  Brendan Gregg   Created this.
```

```
 */

BEGIN
{
        printf("Counting syscalls... Hit Ctrl-C to end.\n");
        // ausyscall --dump | awk 'NR > 1 { printf("\t@sysname[%d] = \"%s\";\n", $1,
$2); }'
}

tracepoint:raw_syscalls:sys_enter
{
        @syscall[args->id] = count();
        @process[comm] = count();
}

END
{
        printf("\nTop 10 syscalls IDs:\n");
        print(@syscall, 10);
        clear(@syscall);

        printf("\nTop 10 processes:\n");
        print(@process, 10);
        clear(@process);
}
```

bpftrace Beispiel-Skripte für Server-Diagnose

- `syscount.bt` protokolliert Systemaufrufe

```
# syscount.bt
Attaching 3 probes...
Counting syscalls... Hit Ctrl-C to end.
^C
Top 10 syscalls IDs:
@syscall[217]: 851
@syscall[4]: 863
@syscall[232]: 999
@syscall[46]: 1532
@syscall[9]: 2069
@syscall[0]: 2366
@syscall[47]: 3532
@syscall[5]: 3573
@syscall[3]: 4147
@syscall[257]: 4883

Top 10 processes:
@process[systemd-fstab-g]: 439
@process[cloud-init-gene]: 540
@process[systemd-detect-]: 732
@process[systemd-machine]: 1018
@process[systemd-gpt-aut]: 1697
@process[dbus-daemon]: 1733
```

```
@process[systemd-logind]: 1812
@process[systemctl]: 2268
@process[systemd-sysv-ge]: 3532
@process[systemd]: 12369
```

- Liste der Linux-Syscalls ausgeben (`ausyscall` ist Teil des Linux Audit-Daemons, Paket `auditd`)

```
    # ausyscall --dump | awk 'NR > 1 { printf("\t@sysname[%d] = \"%s\";\n", $1, $2);
}'
```

- `capable.bt` überwacht Syscalls, welche die Capabilities von Linux-Prozessen ändern:

```
# capable.bt
Attaching 3 probes...
Tracing cap_capable syscalls... Hit Ctrl-C to end.
TIME      UID    PID    COMM             CAP  NAME                 AUDIT
13:05:07  0      17406  ln               21   CAP_SYS_ADMIN        2
[145/1917]
13:05:07  0      295    systemd-journal  21   CAP_SYS_ADMIN        0
13:05:07  0      1      systemd          19   CAP_SYS_PTRACE       2
13:05:07  0      1      systemd          19   CAP_SYS_PTRACE       2
13:05:07  0      1      systemd          19   CAP_SYS_PTRACE       2
13:05:07  0      1      systemd          19   CAP_SYS_PTRACE       2
13:05:07  0      1      systemd          19   CAP_SYS_PTRACE       2
13:05:07  0      1      systemd          1    CAP_DAC_OVERRIDE     0
13:05:07  0      1      systemd          5    CAP_KILL             0
13:05:07  0      1      systemd          1    CAP_DAC_OVERRIDE     0
13:05:07  0      1      systemd          5    CAP_KILL             0
13:05:07  0      17385  systemctl        21   CAP_SYS_ADMIN        2
13:05:07  0      1      systemd          21   CAP_SYS_ADMIN        0
13:05:07  0      1      systemd          21   CAP_SYS_ADMIN        0
13:05:07  0      1      systemd          21   CAP_SYS_ADMIN        0
```

- `bashreadline.bt` instrumentiert die von der `bash` benutzten `readline` Bibliothek und zeichnet alle in der `bash` (systemweit) eingegebenen Befehle auf:

```
# bashreadline.bt
Attaching 2 probes...
Tracing bash commands... Hit Ctrl-C to end.
TIME      PID    COMMAND
13:07:39  11349  bash
13:07:48  17414  su - nutzer
13:08:06  17416  ls
13:08:16  17416  rm -rf *
13:08:20  17416  ls -la
13:08:25  17416  exit
```

bpftrace One-Liner

- Systemaufrufe nach Prozessen

```
bpftrace -e 'tracepoint:raw_syscalls:sys_enter { @[pid, comm] = count(); }'
```

- `execve` Systemaufrufe (neue Prozesse inkl. Argumente)

```
bpftrace -e 'tracepoint:syscalls:sys_enter_execve { join(args->argv); }'
```

- Den Aufrufstapel (Execution-Stack) eines Prozesses (hier PID 189) in fünf Ebenen protokollieren

```
bpftrace -e 'profile:hz:49 /pid == 189/ { @[ustack(5)] = count(); }'
```

- Speicheranforderungen von Prozessen protokollieren (ACHTUNG: verursacht hohe Systemlast)

```
bpftrace -e 'u:/lib/x86_64-linux-gnu/libc.so.6:malloc { @[ustack, comm] = sum(arg0); }'
```

3.18.5. Bücher zu eBPF/BCC

Linux Observability with BPF

- by Lorenzo Fontana, David Calavera
- Publisher: O'Reilly Media, Inc.
- Release Date: November 2019
- ISBN: 9781492050209

BPF Performance Tools: Linux System and Application Observability

- by Brendan Gregg
- Publisher: Addison-Wesley Professional
- Release Date: December 2019
- ISBN: 9780136624523
- http://www.brendangregg.com/bpf-performance-tools-book.html [http://www.brendangregg.com/bpf-performance-tools-book.html]

Systems Performance, 2nd Edition

- by Brendan Gregg
- Publisher: Addison-Wesley Professional
- Release Date: December 2020
- ISBN: 978-0136820154
- https://www.brendangregg.com/systems-performance-2nd-edition-book.html [https://www.brendangregg.com/systems-performance-2nd-edition-book.html]

3.18.6. Links

- A brief introduction to XDP and eBPF https://blogs.igalia.com/dpino/2019/01/07/a-brief-introduction-to-xdp-and-ebpf/ [https://blogs.igalia.com/dpino/2019/01/07/a-brief-introduction-to-xdp-and-ebpf/]
- Dive into BPF: a list of reading material https://qmonnet.github.io/whirl-offload/2016/09/01/dive-

into-bpf/ [https://qmonnet.github.io/whirl-offload/2016/09/01/dive-into-bpf/]

- A curated list of awesome projects related to eBPF: https://github.com/zoidbergwill/awesome-ebpf [https://github.com/zoidbergwill/awesome-ebpf]
- BPF, eBPF, XDP and Bpfilter… What are These Things and What do They Mean for the Enterprise? https://www.netronome.com/blog/bpf-ebpf-xdp-and-bpfilter-what-are-these-things-and-what-do-they-mean-enterprise/ [https://www.netronome.com/blog/bpf-ebpf-xdp-and-bpfilter-what-are-these-things-and-what-do-they-mean-enterprise/]
- BCC Tutorial https://github.com/iovisor/bcc/blob/master/docs/tutorial.md [https://github.com/iovisor/bcc/blob/master/docs/tutorial.md]
- Learn eBPF Tracing: Tutorial and Examples http://www.brendangregg.com/blog/2019-01-01/learn-ebpf-tracing.html [http://www.brendangregg.com/blog/2019-01-01/learn-ebpf-tracing.html]
- Network debugging with eBPF (RHEL 8) https://developers.redhat.com/blog/2018/12/03/network-debugging-with-ebpf/ [https://developers.redhat.com/blog/2018/12/03/network-debugging-with-ebpf/]
- Using eBPF for network traffic analysis https://www.ntop.org/wp-content/uploads/2018/10/Sabella.pdf [https://www.ntop.org/wp-content/uploads/2018/10/Sabella.pdf]
- BCC Referenz: https://github.com/iovisor/bcc/blob/master/docs/reference_guide.md [https://github.com/iovisor/bcc/blob/master/docs/reference_guide.md]
- eBPF Summit 2021:
  - Tag 1: https://www.youtube.com/watch?v=Kp3PHPuFkaA [https://www.youtube.com/watch?v=Kp3PHPuFkaA]
  - Tag 2: https://www.youtube.com/watch?v=ZNtVedFsD-k [https://www.youtube.com/watch?v=ZNtVedFsD-k]
- eBPF Foundation https://ebpf.io/ [https://ebpf.io/]
- Comparing SystemTap and bpftrace https://lwn.net/Articles/852112/ [https://lwn.net/Articles/852112/]
- nsjail | A light-weight process isolation tool, making use of Linux namespaces and seccomp-bpf syscall filters (with help of the kafel bpf language) [https://nsjail.dev/#usrbinfind-in-a-minimal-file-system-only-usrbinfind-accessible-from-usrbin]
- How io_uring and eBPF Will Revolutionize Programming in Linux - The New Stack [https://thenewstack.io/how-io_uring-and-ebpf-will-revolutionize-programming-in-linux/]
- Auch in eBPF kann es Sicherheits-Fehler geben Zero Day Initiative — CVE-2020-8835: Linux Kernel Privilege Escalation via Improper eBPF Program Verification [https://www.thezdi.com/blog/2020/4/8/cve-2020-8835-linux-kernel-privilege-escalation-via-improper-ebpf-program-verification]
- boopkit – Linux eBPF backdoor over TCP. Spawn reverse shells, RCE, on prior privileged access. https://github.com/kris-nova/boopkit [https://github.com/kris-nova/boopkit]
- BPFDoor — an active Chinese global surveillance tool https://doublepulsar.com/bpfdoor-an-active-chinese-global-surveillance-tool-54b078f1a896?gi=bd3c470d93a8 [https://doublepulsar.com/bpfdoor-an-active-chinese-global-surveillance-tool-54b078f1a896?gi=bd3c470d93a8]
- hBPF = eBPF in hardware (An extended Berkley Packet Filter CPU written entirely in Python3 for PC and FPGA) https://github.com/rprinz08/hBPF [https://github.com/rprinz08/hBPF]

3.18.7. eBPF Vortrag vom Security–Forum 2024

- https://ebpf.linux–sicherheit.org/ [https://ebpf.linux–sicherheit.org/]

3.19. BIND 9 ALS DNS–CACHE MIT DNSSEC

- Wir arbeiten auf dem VM–Server
- BIND 9 installieren und Version prüfen (BIND 9 sollte > 9.16.x sein)

```
# apt install bind9
# named -V
```

- Konfiguration anpassen (Datei `/etc/bind/named.conf.options`)

```
options {
  directory "/var/cache/bind";
  dnssec-validation auto;    # Default seit BIND 9.16
};
```

- BIND 9 neu starten

```
# systemctl restart bind9
# systemctl status bind9
```

- Testen (AD–Flag muss erscheinen)

```
# dig @localhost bund.de +dnssec +multi
```

3.20. UNBOUND DNS–RESOLVER MIT DNSSEC

- wir arbeiten auf der Debian VPS (als Benutzer `root` per `sudo -s`)
- Unbound und die DNS–Tools installieren

```
apt install unbound dnsutils
```

- DNS–Abfrage manuell testen, bei einer bekannt DNSSEC geschützten Domain muss im Header das `AD`–Flag (Authentic Data) angezeigt werden. Das `AD`–Flag ist das Signal, das die DNS–Daten DNSSEC abgesichert sind **und** korrekt überprüft (validiert) wurden:

```
dig @127.0.0.1 linuxhotel.de a
```

- die Linux–Resolver Konfiguration anpassen, damit alle Anwendungen des Servers über den Unbound DNS–Resolver gehen

```
echo "nameserver 127.0.0.1" > /etc/resolv.conf
```

- DNSSEC Vertrauenskette mit dem Tool `drill` aus dem Paket `ldnsutils` anzeigen

```
apt install ldnsutils
```

- den Root–KSK Schlüssel besorgen und lokal abspeichern

```
dig @a.root-servers.net dnskey . | grep 257 > root.key
```

- DNSSEC Vertrauenskette prüfen

```
drill -SD -k root.key www.linuxhotel.de a
```

- Vergleiche die Ausgabe von `drill` mit der Ausgabe der Webseite `https://dnsviz.net` für die Domain `linuxhotel.de`