LINUX SERVER SICHERHEIT KURSNOTIZEN SEPTEMBER 2024 (ROCKY LINUX 9)

CARSTEN STROTMANN

Version 2.2, 28.09.2024

INHALTSVERZEICHNIS

- Kursnotizen: http://notes.defaultroutes.de [http://notes.defaultroutes.de]
- Literaturliste
  - SUDO Mastery https://www.michaelwlucas.com/nonfiction/sudo-mastery [https://www.michaelwlucas.com/nonfiction/sudo-mastery]
  - SSH Mastery https://www.michaelwlucas.com/nonfiction/ssh-mastery [https://www.michaelwlucas.com/nonfiction/ssh-mastery]
  - 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 [http://www.cl.cam.ac.uk/<sub>rja14/book.html])
  - Thinking Security: Stopping Next Year's Hackers (Addison-Wesley Professional Computing) ISBN-10: 0134277546/ISBN-13: 978-0134277547
  - Nikolai Philipp Tschacher□—□Typosquatting in Programming Language Package Managers http://incolumitas.com/data/thesis.pdf [http://incolumitas.com/data/thesis.pdf]
  - If It's Smart, It's Vulnerable by Mikko Hypponen https://www.ifitssmartitsvulnerable.com/ [https://www.ifitssmartitsvulnerable.com/]

CHAPTER 1. TAG 1

1.1. 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. Nachfolgend stellen wir die Anzahl der Runden auf 1.000.000 (1 Million) ein.

1.1.1. Sicherheit der Benutzerpasswörter

- In der Datei `/etc/pam.d/system-auth` 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
[...]
```

- `yescrypt` Passwörter werten das Feld logarithmisch aus und speichern anders ab

```
[...]
password    [success=1 default=ignore]    pam_unix.so obscure yescrypt rounds=11
[...]
```

- `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+, Kali Linux 2021.1+ und Ubuntu 22.04+. Außerdem wird es von Fedora 29+ und RHEL 9+ unterstützt. 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.
- `yescrypt` Dokumentation: https://github.com/openwall/yescrypt/blob/main/README [https://github.com/openwall/yescrypt/blob/main/README]

1.1.2. Gruppenpasswörter

- Unix/Linux Systeme erlauben Gruppenpasswörter. Ein Gruppenpasswort erlaubt es einem oder mehreren Benutzern temporär die Gruppenzugehörigkeit zu wechseln, ohne permanentes Mitglied einer Gruppe zu sein
- Ein Benutzer kann mittels des Befehls `newgrp` die Gruppenzugehörigkeit wechseln

- Die Gruppenpasswörter werden mit dem Befehl `gpasswd` gesetzt und verwaltet
- Gruppenpasswörter werden in den Dateien `/etc/group` und `/etc/gshadow` gespeichert. `/etc/group` enthält Gruppeninformationen und `/etc/gshadow` enthält das Passwort (als Hash)
- Die Sicherheit der Gruppenpasswörter werden in der Datei `/etc/login.defs` festgelegt. Wenn Gruppenpasswörter benutzt werden, wird empfolen diese Werte mit der PAM-Konfiguration für Account-Passwörter gleich zu halten.

```
ENCRYPT_METHOD SHA512
SHA_CRYPT_MIN_ROUNDS 500000
SHA_CRYPT_MAX_ROUNDS 1000000
```

- Das Programm `sudo` (oder ähnlichen Erweiterungen des Linux-Systems) kann die Gruppenpasswörter ersetzen und bietet in der Regel eine bessere Kontrolle über die Rechteerweiterungen der Benutzer und ist daher den Gruppenpasswörtern vorzuziehen

1.1.3. 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)

ID	Methode
6	SHA-512 (seit glibc 2.7)
y/7	yescrypt (Debian 11/RHEL 9/Fedora 35/Ubuntu 22.04)

1.2. PASSWORT-RICHTLINIEN / ABLAUFENDE PASSWÖRTER (CHAGE)

- Mit dem Befehl `chage` können die Informationen zur Passwort-Gültigkeitsdauer verwaltet werden.
- Der Parameter `-l` gibt die Passwort-Ablauf-Informationen eines Benutzers aus:

```
useradd -m -N fritz
passwd fritz
chage -l fritz
```

- Mittels des Parameters `-E 0` kann ein Account-Passwort als abgelaufen markiert werden

```
chage -E 0 fritz
```

- Hier ist der Account ab einem bestimmten Datum nicht mehr gültig

```
chage -E 2025-03-01 fritz
```

- Die Ablaufzeit zurücksetzen (entfernen)

```
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
chage -d 0 fritz                      # Passwort-Änderung erzwingen
```

1.3. BEFEHL SU

- mit dem Programm `su` (*Switch User* eigentlich *Substitute User*) kann ein Benutzer in 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

1.4. BEFEHL 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

```
dnf install sudo
```

- `sudo` Basis-Konfiguration ausgeben

```
sudo -V
```

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

```
EDITOR=emacs visudo
```

1.4.1. sudo und das Environment

- Der Befehl `sudo` löscht oder ersetzt kritische Umgebungs-Variablen, welche (bekannte) Auswirkungen auf die Sicherheit von Kommandos und Programmen haben.
- Der Befehl `sudo -V` zeigt an, welche Umgebungsvariablen entfernt, geprüft oder neu gesetzt werden:

```
# sudo -V
[...]
Ignore '.' in $PATH
[...]
```

```
Always set $HOME to the target user's home directory
[...]
Environment variables to check for safety:
        TZ
        TERM
        LINGUAS
        LC_*
        LANGUAGE
        LANG
        COLORTERM
Environment variables to remove:
        *=()*
        RUBYOPT
        RUBYLIB
        PYTHONUSERBASE
        PYTHONINSPECT
        PYTHONPATH
        PYTHONHOME
        TMPPREFIX
        ZDOTDIR
        READNULLCMD
        NULLCMD
        FPATH
        PERL5DB
        PERL5OPT
        PERL5LIB
        PERLLIB
        PERLIO_DEBUG
        JAVA_TOOL_OPTIONS
        SHELLOPTS
        BASHOPTS
        GLOBIGNORE
        PS4
        BASH_ENV
        ENV
        TERMCAP
        TERMPATH
        TERMINFO_DIRS
        TERMINFO
        _RLD*
        LD_*
        PATH_LOCALE
        NLSPATH
        HOSTALIASES
        RES_OPTIONS
        LOCALDOMAIN
        CDPATH
        IFS
Environment variables to preserve:
        XAUTHORITY
        _XKB_CHARSET
        LINGUAS
        LANGUAGE
        LC_ALL
```

```
        LC_TIME
        LC_TELEPHONE
        LC_PAPER
        LC_NUMERIC
        LC_NAME
        LC_MONETARY
        LC_MESSAGES
        LC_MEASUREMENT
        LC_IDENTIFICATION
        LC_COLLATE
        LC_CTYPE
        LC_ADDRESS
        LANG
        USERNAME
        QTDIR
        PS2
        PS1
        MAIL
        LS_COLORS
        KDEDIR
        HISTSIZE
        HOSTNAME
        DISPLAY
        COLORS
[...]
```

`sudo -s`, `sudo -i` vs. `sudo su -`

- Der Befehl `sudo -s` erzeugt eine Root-Shell mit der in der Variablen `$SHELL` konfigurierten Shell des (Quell-)Benutzers
- Der Befehl `sudo -i` erzeugt eine neue Login-Shell mit der Umgebung des Ziel-Benutzers (`.profile` und `.bashrc` des Ziel-Benutzers werden geladen)
- Die Kombination `sudo su -` ist eine alte Form aus den Zeiten, in denen es `sudo -i` noch nicht gab. Diese Kombination erzeugt eine neue Root-Login-Shell (`.profile` und `.bashrc` des Root-Benutzers werden geladen). Diese Kombination kann heute durch `sudo -i` ersetzt werden.

1.4.2. sudo Konfiguration Beispiel:

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

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

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

1.4.3. Aufgabe:

- Erstelle eine `sudo` Konfiguration, um es dem Benutzer `user` zu erlauben, die Logdateien `/var/log/messages` und `/var/log/kdump.log` unter den Benutzerberechtigungen des Benutzers `root` mit den Programmen `more` und `less` anzuschauen

Test:

- in einem anderen Terminal/TMUX-Fenster, teste `sudo` als `user`

```
user$ sudo more /var/log/messages
user$ sudo less /var/log/kdump.log
user$ sudo less /etc/shadow       # <--- darf nicht funktionieren
user$ sudo more /var/log/cron     # <--- darf nicht funktionieren
```

1.4.4. Lösungs-Beispiel

```
visudo -f /etc/sudoers.d/log-message-view
------
...
user  ALL=(root) /usr/bin/more /var/log/messages
user  ALL=(root) /usr/bin/more /var/log/kdump.log
user  ALL=(root) /usr/bin/less /var/log/messages
user  ALL=(root) /usr/bin/less /var/log/kdump.log
...
```

Frage:

- Gibt es mit dieser Konfiguration ein Sicherheitsproblem?

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/kdump.log
```

1.4.5. sudo Aliases

- In der `sudo` Konfigurationen können *Alias* Definitionen die Komplexität der Konfiguration verringern. Aliase erlauben es mehrere Rechner, Benutzer oder Kommandos hinter einem sprechendem Namen zusammenzufassen

```
  # 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
```

1.4.6. Dateien editieren mit sudoedit

- Benutzer `user` soll die Datei `/etc/rsyslog.conf` editieren dürfen. Diese Datei darf nur mit Rechten des Benutzers `root` geschrieben werden. Bei der Benutzung von `sudoedit` wird die Datei unter Root–Rechten in eine temporäre Datei kopiert und die Datei wird mit einem Editor unter den Rechten des Benutzers geöffnet. Herbei werden die Sicherheitsrisiken durch die Ausführung von *komplexen* Editoren wie `vim` oder `emacs` (z.B. Shell–Funktionen der Editoren) vermieden
- In der Datei `/etc/sudoers`

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

1.4.7. sudo replay

- Die Funktion `sudoreplay` erlaubt es die Ein–und Ausgabe einer Sudo–Sitzung mitzuschneiden und zu einem späteren Zeitpunkt wieder abzuspielen.
- Füge die folgenden Konfigurationszeilen in die `/etc/sudoers` Datei ein

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

- **nutzerXX darf root werden**

```
nutzerXX ALL=(root) ALL
```

- **Benutze sudo als nutzerXX um eine interaktive Shell zu bekommen**

```
nutzerXX$ sudo -s
```

- **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/
```

1.4.8. Einfache Intrusion Detection mit sudo (ab sudo 1.8.7)

- **In der sudo Konfigurationsdatei können Programme mittels eines kryptografischen Hash gesichert werden. Das sudo Programm prüft vor einem Aufruf mit sudo ob der Hash der Programmdatei noch mit dem in der Konfiguration hinterlegten Wert übereinstimmt und verhindert bei einem Unterschied die Ausführung des Befehls**
- **Mit dieser Funktion lässt sich eine einfache Intrustion-Detection (z.B. zum Aufspüren von Root-Kits) implementieren. Jedoch müssen nach jedem Update des Systems ggf. die Hashes in der Sudo-Konfigurationsdatei aktualisiert werden**

```
 openssl dgst -sha256 /usr/bin/passwd
 SHA256(/usr/bin/passwd)=
a92b1b6fb52549ed23b12b32356c6a424d77bcf21bfcfbd32d48e12615785270
 visudo
 ----
 nutzerXX ALL= sha256:a92b1b6fb52... /usr/bin/passwd
 ----
```

1.4.9. sudo Konfiguration (Passwort Cache Beispiele)

- **Sudo kann ein eingegebenes Passwort für eine bestimmte Zeit im Speicher zwischenspeichern. Hierdurch werden wiederholte Passwort-Eingaben bei mehreren Sudo-Befehlen vermieden.**
- **Das Caching-Verhalten von sudo kann in der Sudo-Konfigurationsdatei eingestellt werden:**

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

1.4.10. Befehle ohne Passwort

- `sudo` kann Befehle ohne die Eingabe eines Passworts ausführen. Dies ist speziell für automatische Skripte welche z.B. via SSH ausgeführt werden interessant.

```
nutzer ALL=(dba) NOPASSWD: /opt/oracle/bin/befehl
```

- Passwortlose `sudo` Befehle können ein Sicherheitsrisiko sein und sollten auf bestimmte Befehle begrenzt werden. Automatisch via SSH ausgeführte `sudo` Befehle sollten auch auf der SSH-Seite durch ACLs begrenzt werden (siehe Kapitel zu `ssh`).

1.4.11. Passwort des Zielaccount angeben (SUSE Linux Default)

- Einige Linux-Distributionen (z.B. SUSE Linux) verlangen bei der Benutzung von `sudo` das Passwort dies Ziel-Accounts (z.B. `root`) anstatt des Passwort des Quell-Accounts (des Benutzers).

```
Defaults targetpw
```

- In dieser Konfiguration arbeitet `sudo` funktionell ähnlich wie der `su` Befehl

1.5. PAM – PLUGGABLE AUTHENTICATION MODULES

- Dokumentation des Linux-PAM-Systems 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]
- Red Hat/Rocky 8/9 Pfad für PAM-Module `/lib64/security/`
- Beispiel der PAM Konfigurationsdatei `/etc/pam.d/system-auth`

```
cat /etc/pam.d/system-auth

session     optional      pam_keyinit.so revoke
session     required      pam_limits.so
-session     optional      pam_systemd.so
session     [success=1 default=ignore] pam_succeed_if.so service in crond quiet use_uid
session     required      pam_unix.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 = Modul wird ausgeführt, 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

1.5.1. 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 **login** konfiguriert, aber nicht aktiv (Datei `/etc/pam.d/login`)
- Lese die Man-Pages der PAM-Module für den Dienst **login** der **account**-Funktionen, finde heraus, welches Modul das Login aller Nicht-Root-Benutzer verweigern kann, und wie es aktiviert wird.
- Aktiviere diese Funktion und teste diese aus. Wechsle mit STRG+ALT+F2 auf die Text-Konsole und versuche Dich dort mit dem normalen Benutzer anzumelden. **Alternativ**: Anmeldung per SSH über das Loopback-Interface:

```
ssh user@localhost
```

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

1.5.2. Lösung:

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

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

1.5.3. Aufgabe: 2-Faktor Authentisierung - OATH Open Authentication Event Token (HOTP)

- Bei RedHat EL System befindet sich das OATH Modul in den EPEL Repositories (Extra Pakets for Enterprise Linux). Das EPEL Repository kann über das Paket `epel-release` aktiviert werden (ggf. Richtlinien zur Installation von Programmen aus externen Quellen beachten)

```
dnf install epel-release
```

- 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

```
dnf install pam_oath oathtool
```

- Wir erstellen einen neuen Benutzer, an welchem wir die Funktion testen können

```
root$ adduser -m nutzer
root$ passwd nutzer
```

- `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 inkl. Seed-Werte um den Startwert zu setzen

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

- Beispiel: Passwort in HEX-Zahl (Seed) umrechnen:

```
echo "villa" | od -x
0000000 6976 6c6c 0a61
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 - nutzer
```

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

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

1.5.5. Weitere Links zu PAM

- Understanding the effects of PAM module results ('controls' in PAM jargon) https://utcc.utoronto.ca/}cks/space/blog/sysadmin/PAMModuleResultsEffects [https://utcc.utoronto.ca/<sub>cks/space/blog/sysadmin/PAMModuleResultsEffects]
- A Linux PAM setup and the problem of stopping authentication https://utcc.utoronto.ca/}cks/space/blog/linux/PAMStackingAndStopping [https://utcc.utoronto.ca/<sub>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]

1.6. KERNEL PARAMETER

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

1.6.1. 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

1.6.2. AppArmor

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

- AppArmor an/ausschalten

1.6.3. 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

1.6.4. Signierte Module

```
module.sig_enforce
```

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

1.6.5. NOEXEC

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

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

1.6.6. Datei-Capabilities

```
no_file_caps
```

- Schaltet die Datei-Capabilities aus

1.6.7. Keine Module nachladen

```
nomodule
```

- es können keine Module nachgeladen werden

1.6.8. Kernel-Panic

```
panic=<timeout>
```

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

1.6.9. sysctl

- Übersicht- `sysctl` Parameter-Tuning: https://klaver.it/linux/sysctl.conf [https://klaver.it/linux/sysctl.conf]

```
sysctl -a
```

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

```
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
```

1.6.10. IPv6 sysctl Empfehlungen für Server

- Empfehlungen für Linux-Server mit dynamischer (SLAAC) Adressen-Konfiguration
- Bei Servern mit statischer Adressen-Konfigurationen können viele IPv6 Funktionen ausgeschaltet werden, um die Angriffsfläche zu verringern. Siehe https://ernw.de/download/ERNW_Guide_to_Securely_Configure_Linux_Servers_For_IPv6_v1_0.pdf [https://ernw.de/download/ERNW_Guide_to_Securely_Configure_Linux_Servers_For_IPv6_v1_0.pdf]

```
net.ipv6.conf.all.use_tempaddr = 0
net.ipv6.conf.all.accept_ra_mtu = 1
net.ipv6.conf.all.accept_ra_rtr_pref = 1
net.ipv6.conf.all.accept_ra_rtr_pref = 1
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.all.accept_untracked_na = 0
```

```
net.ipv6.conf.all.addr_gen_mode = 0
net.ipv6.conf.all.autoconf = 0
net.ipv6.conf.all.enhanced_dad = 1
net.ipv6.conf.all.max_addresses = 4
net.ipv6.conf.all.mc_forwarding = 0
net.ipv6.conf.all.mtu = 1280
net.ipv6.conf.all.optimistic_dad = 0
net.ipv6.conf.all.suppress_frag_ndisc = 1

net.ipv6.conf.default.use_tempaddr = 0
net.ipv6.conf.default.accept_ra_mtu = 1
net.ipv6.conf.default.accept_ra_rtr_pref = 1
net.ipv6.conf.default.accept_ra_rtr_pref = 1
net.ipv6.conf.default.accept_redirects = 0
net.ipv6.conf.default.accept_source_route = 0
net.ipv6.conf.default.accept_untracked_na = 0
net.ipv6.conf.default.addr_gen_mode = 0
net.ipv6.conf.default.autoconf = 0
net.ipv6.conf.default.enhanced_dad = 1
net.ipv6.conf.default.max_addresses = 4
net.ipv6.conf.default.mc_forwarding = 0
net.ipv6.conf.default.mtu = 1280
net.ipv6.conf.default.optimistic_dad = 0
net.ipv6.conf.default.suppress_frag_ndisc = 1

net.ipv6.icmp.ratelimit = 1000
net.ipv6.icmp.ratemask = 0-1,3-127

net.ipv6.ip6frag_high_thresh = 4194304
net.ipv6.ip6frag_low_thresh = 3145728
net.ipv6.ip6frag_secret_interval = 0
net.ipv6.ip6frag_time = 60

net.ipv6.route.max_size = 4096
net.ipv6.route.min_adv_mss = 1220
net.ipv6.route.mtu_expires = 600
```

1.7. EBPF

1.7.1. Was ist eBPF, XDP und BCC?

- eBPF ist die *extended Berkeley Packet Filter* virtuelle Maschine im Linux Kernel
- XDP (eXpress Data Path) ist eine eBPF Schnittstelle zum Netzwerkstack
- BCC ist die *BPF Compiler Collection*, eine Sammlung von Tools und eBPF Programmen
- eBPF ist eine Weiterentwicklung der originalen Berkeley Packet Filter Technologie https://en.wikipedia.org/wiki/Berkeley_Packet_Filter [https://en.wikipedia.org/wiki/Berkeley_Packet_Filter]

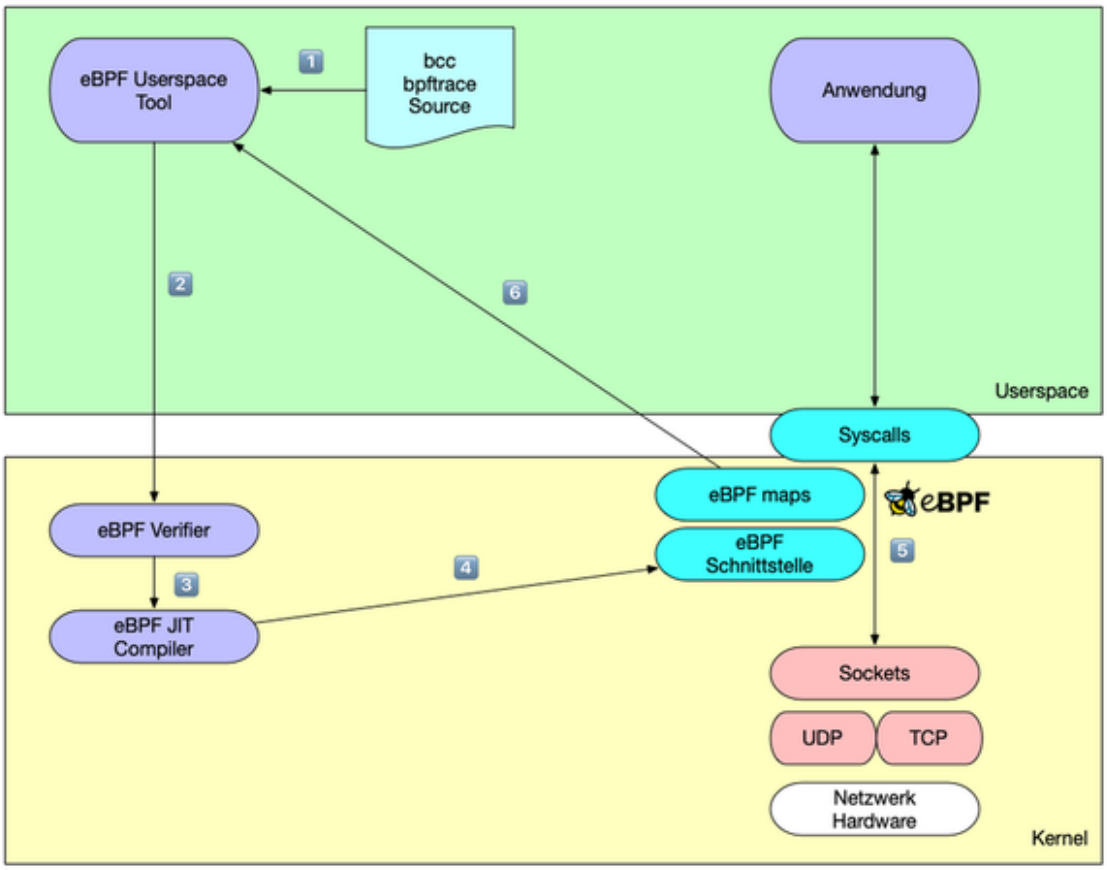
1.7.2. Die eBPF Idee

- eBPF erlaubt es dem Benutzer, Programme im Betriebssystem–Kern innerhalb einer Sandbox

auszuführen

- eBPF ermöglicht es, die Funktionen des Betriebssystem–Kerns sicher und effizient zu erweitern, ohne den Quell–Code des Kernels ändern oder Module laden zu müssen
- eBPF Programme können Netzwerk–Pakete (und andere Datenstrukturen) innerhalb des Linux–Kernels überwachen und verändern
- eBPF Programme sind **keine** Kernel Module, es ist nicht notwendig ein Kernel–Entwickler zu sein um eBPF benutzen zu können
  - Wissen über Programmierung in der Sprache "C" ist jedoch von Vorteil

1.7.3. eBPF



1.7.4. eBPF Einsatzgebiete

- Einsatzgebiete für eBPF
  - Netzwerk Sicherheit (erweiterte Firewall Funktionen)
  - Host / Container Security
  - Forensische Analyse laufender Prozesse

- Fehlerdiagnose
- Geschwindigkeitsmessungen
- Rootkits? Backdoors im Kernel oder in Netzwerkkarten?

1.7.5. eBPF Verfügbarkeit

- eBPF ist in modernen Linux Systemen verfügbar (ab Kernel 3.18+) und wird derzeit von Microsoft auf das Windows Betriebssystem portiert
  - Blog Post "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/]

1.7.6. Die Wurzeln von BPF

- Der originale BSD Packet Filter (BPF) wurde von Steven McCanne und Van Jacobson am Lawrence Berkeley Laboratory entwickelt (https://www.tcpdump.org/papers/bpf-usenix93.pdf [https://www.tcpdump.org/papers/bpf-usenix93.pdf])
  - BPF wurde auf fast alle Unix/Linux Systeme und viele non-Unix Betriebssysteme portiert (z.B. Windows, BeOS/Haiku, OS/2 ...)
  - BPF ist die Basis-Technologie hinter bekannten Netzwerk-Sniffing Werkzeugen wie `tcpdump` und *Wireshark*

1.7.7. BPF am Beispiel von tcpdump

- Ein BPF-Filter (z.B. für *tcpdump*) wird in einen Bytecode für die BPF virtuelle Maschine im Linux-Kernel übersetzt und in den Kernel geladen
  - Das Betriebssystem ruft das BPF-Programm für jedes Netzwerk-Paket auf, welches den Netzwerk-Stack durchläuft
  - Nur Pakete, welche auf den Filter-Ausdruck passen, werden an das Programm im Userspace weitergeleitet (der `tcpdump` Prozess in dieses Beispiel)
  - BPF reduziert die Menge der Daten, welche zwischen dem Kernel und dem Userspace ausgetauscht werden müssen

1.7.8. BPF am Beispiel von tcpdump

`tcpdump` kann angewiesen werden den BPF Quellcode des `tcpdump` Filters auszugeben:

```
# tcpdump -d port 53 and host 1.1.1.1
Warning: assuming Ethernet
(000) ldh      [12]
(001) jeq      #0x86dd          jt 19   jf 2
(002) jeq      #0x800           jt 3    jf 19
(003) ldb      [23]
(004) jeq      #0x84            jt 7    jf 5
(005) jeq      #0x6             jt 7    jf 6
(006) jeq      #0x11            jt 7    jf 19
(007) ldh      [20]
```

```
(008) jset      #0x1fff          jt 19   jf 9
(009) ldxb      4*([14]&0xf)
(010) ldh       [x + 14]
(011) jeq       #0x35            jt 14   jf 12
(012) ldh       [x + 16]
(013) jeq       #0x35            jt 14   jf 19
(014) ld        [26]
(015) jeq       #0x1010101       jt 18   jf 16
(016) ld        [30]
(017) jeq       #0x1010101       jt 18   jf 19
(018) ret       #262144
(019) ret       #0
```

1.7.9. eBPF vs. BPF

- Während BPF (heute auch cBPF = classic BPF genannt) Netzwerk Pakete im Betriebssystem–Kern filtert, kann eBPF auf weitere Kernel–Datenstrukturen Filter anwenden und Programm–Code ausführen:
  - Kernel Systemcalls
  - Kernel Tracepoints
  - Kernel Funktionen
  - Userspace Tracepoints
  - Userspace Funktionen

1.7.10. eBPF und der Linux Kernel

- Die erste Version von eBPF wurde im Linux Kernel 3.18 eingeführt
  - Neue Kernel Versionen bringen weitere, neue eBPF Funktionen
  - Linux Distributionen (Red Hat/Canonical/Suse) haben zum Teil eBPF Funktionen auf ältere LTS/EL Kernel Versionen zurückportiert
  - Eine Übersicht der eBPF Funktionen nach Linux Kernel Version aufgeschlüsselt: https://github.com/iovisor/bcc/blob/master/docs/kernel–versions.md [https://github.com/iovisor/bcc/blob/master/docs/kernel–versions.md]

1.7.11. Die eBPF Architektur

Die eBPF virtuelle Maschine

- eBPF Programme werden für eine virtuelle CPU Architektur übersetzt
- Der Programmcode wird in den Linux Kernel geladen und dort geprüft
- Auf populären CPU Architekturen (amd64, AARCH64) wird der eBPF Bytecode in nativen Maschinencode re–compiliert (Just in Time Compiler = JIT)

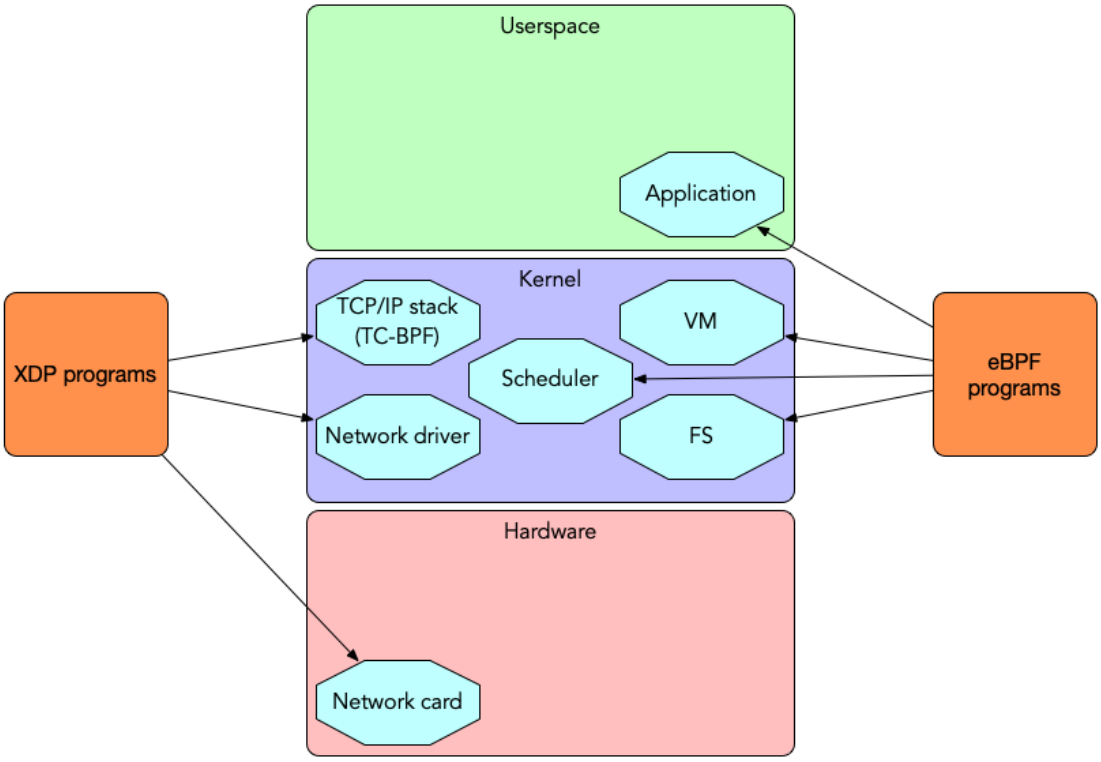
XDP – Express Data Path

- Der *express data path* (XDP) innerhalb des Linux–Kernels ist eine Infrastruktur, um auf unterster Ebene Kontrolle über Netzwerk–Pakete auszuüben
  - Der normale Datenfluss im Linux Netzwerk–Stack kann via XDP umgangen werden
  - eBPF Programme können in den eXpress Data Path (XDP) geladen werden

XDP / eBPF Hardware Offloading

- XDP eBPF Programm können auf verschiedenen Ebenen in den Linux Kernel geladen werden
  - **Offload XDP**: direkt in die Netzwerk–Hardware (ASIC/FPGA, benötigt Unterstützung für XDP in der Hardware, z.B. vorhanden in den Netronome Netzwerkadaptern)
  - **Native XDP**: In den Linux Kernel Netzwerk–Treiber der Netzwerkschnittstelle (benötigt Unterstützung durch den Treiber)
  - **Generic XDP**: In den Linux Kernel Netzwerk–Stack (weniger Performance, aber ohne besondere Unterstützung von Hardware oder Treibern möglich)

XDP / eBPF Ausführungs–Ebenen

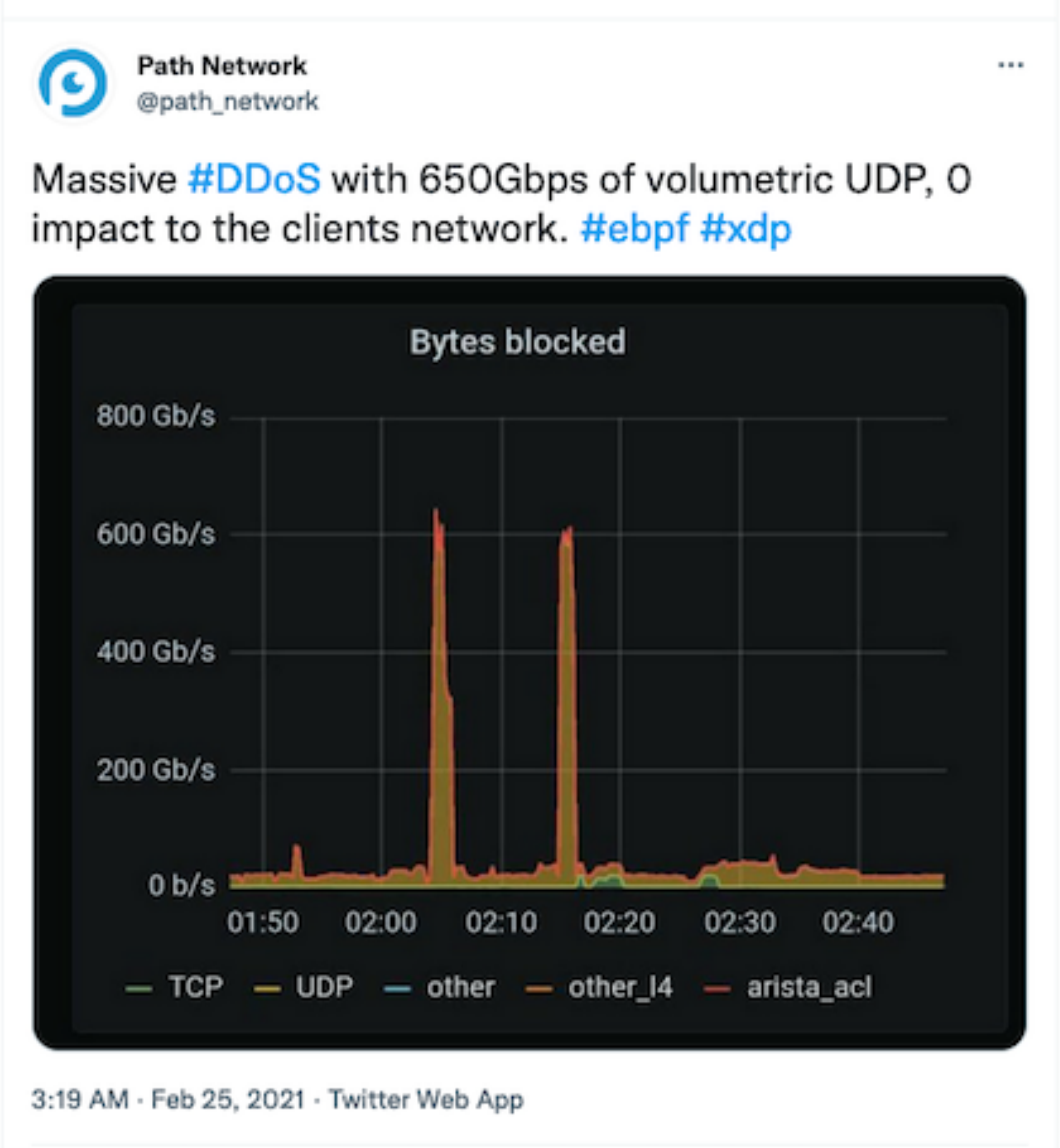


XDP Funktionen

- XDP Programme können
  - **lesen**: Netzwerk–Pakete und Statistiken sammeln
  - **verändern**: Den Inhalt der Netzwerkpakete ändern
  - **verwerfen**: Ausgewählte Netzwerk–Pakete können verworfen werden (Firewall)
  - **umleiten**: Netzwerkpakete können auf die gleichen oder andere Netzwerkschnittstellen umgeleitet werden (Switching/Routing)
  - **durchlassen**: Das Netzwerkpaket wird an den Linux TCP/IP Stack zur normalen Bearbeitung übergeben

XDP vs DDoS Angriffe

- XDP kann unerwünschten Netzwerk–Verkehr schon sehr früh im Netzwerk–Stack verwerfen (z.B. innerhalb der Netzwerk–Hardware). Dies kann zum Schutz gegen DDoS Angriffe eingesetzt werden

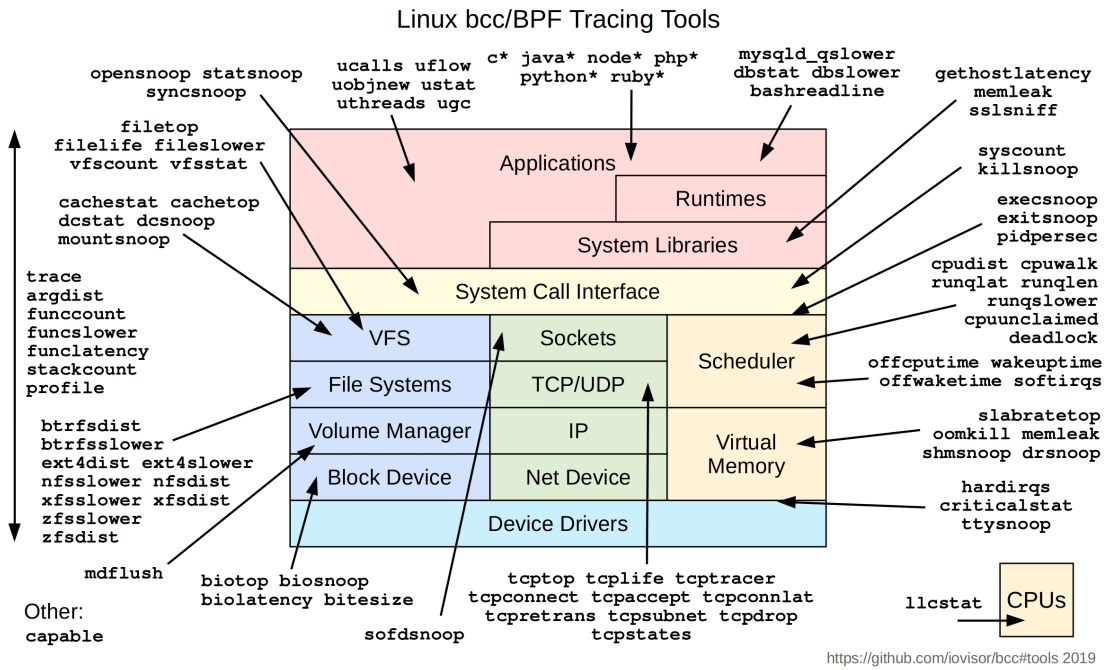


1.7.12. eBPF "Real-World" Anwendungen

eBPF/XDP Support in DNS Software

- Der Open-Source DNS Load-Balancer DNSdist [https://dnsdist.org] von PowerDNS kann DNS Pakete via eBPF und XDP filtern oder per Rate-Limiting beschränken
- Der Knot Resolver (seit Version 5.2.0) kann mittels ePBF und XDP den Linux TCP/IP Stack für DNS Pakete umgehen und die DNS-Pakete direkt an den Knot DNS Resolver Prozess im Userspace weiterleiten (https://knot-resolver.readthedocs.io/en/stable/daemon-bindings-net_xdpsrv.html [https://knot-resolver.readthedocs.io/en/stable/daemon-bindings-net_xdpsrv.html]). Hierdurch wird eine enorme Geschwindigkeitssteigerung der DNS Antwortrate erziehlt.

eBPF Programme zur Analyse des Laufzeitverhalten von Linux-Systemen



https://github.com/iovisor/bcc#tools 2019

Beispiel: eBPF Syscall Auswertung

- Die Syscalls eines BIND 9 Prozesses auswerten mit dem Programm `syscount`

```
# syscount-bpfcc -p `pgrep named` -i 10
Tracing syscalls, printing top 10... Ctrl+C to quit.
[07:34:19]
SYSCALL                   COUNT
futex                       547
getpid                      121
sendto                      113
read                         56
write                        31
epoll_wait                   31
openat                       23
close                        20
epoll_ctl                    20
recvmsg                      20
```

Beispiel: eBPF Prozess Capabilities

- Die Linux Capabilities von laufenden Prozessen anzeigen

```
# capable-bpfcc | grep named
07:36:17  0      29378  (named)          24   CAP_SYS_RESOURCE     1
07:36:17  0      29378  (named)          24   CAP_SYS_RESOURCE     1
07:36:17  0      29378  (named)          12   CAP_NET_ADMIN        1
07:36:17  0      29378  (named)          21   CAP_SYS_ADMIN        1
```

```
07:36:17  0      29378  named            6    CAP_SETGID          1
07:36:17  0      29378  named            6    CAP_SETGID          1
07:36:17  0      29378  named            7    CAP_SETUID          1
07:36:17  109    29378  named            24   CAP_SYS_RESOURCE    1
```

Beispiel: gethostlatency

- Das BCC Programm `gethostlatency` misst die Latenz der client-seitigen DNS Namensauflösung durch Systemaufrufe wie `getaddrinfo` oder `gethostbyname`

```
# gethostlatency-bpfcc
TIME      PID    COMM                  LATms HOST
10:21:58  19183  ping                 143.22 example.org
10:22:18  19184  ssh                    0.03 host.example.de
10:22:18  19184  ssh                   60.59 host.example.de
10:22:35  19185  ping                  23.44 isc.org
10:22:49  19186  ping                4459.72 yahoo.co.kr
```

Sicherheitsanwendungen

- Audit-Frameworks für Linux benutzen eBPF um Prozesse auf einem Host oder im Container *ganzheitlich* zu überwachen:
  - Dateisystemzugriffe
  - Netzwerkverkehr
  - Prozess-Execution
  - Systemcalls

Sicherheitsanwendungen

- Dabei wird jedem Prozess eine *execution-id* zugewiesen und alle Log-Einträge können mittels der ID einem Prozess zugeordnet werden
- eBPF kann direkt Policy-Entscheidungen von Linux-Security-Modulen (LSMs wie SELinux, AppArmor, Tomoya) anpassen
  - eBPF kann direkt Dateizugriffe, Netzwerk-Kommunikation oder Syscalls unterbinden

Sicherheitsanwendungen: Tetragon

Tetragon

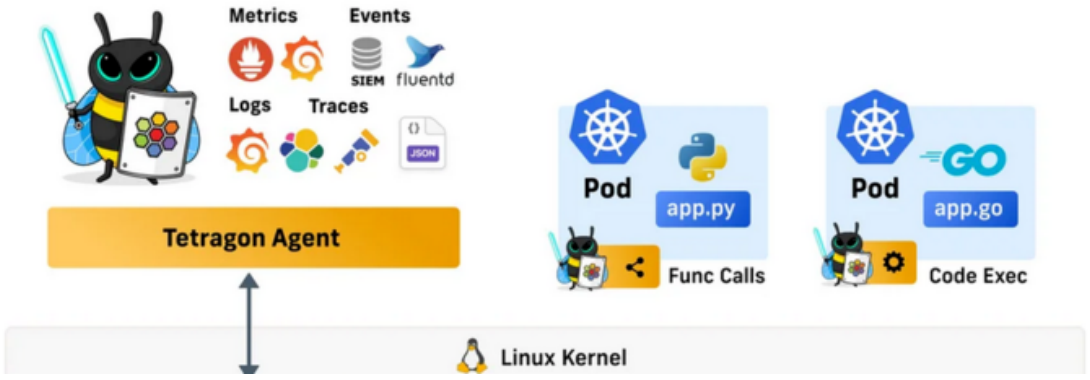
License Apache 2.0

Cilium's new Tetragon component enables powerful realtime, eBPF-based Security Observability and Runtime Enforcement.

Tetragon detects and is able to react to security-significant events, such as

- Process execution events
- System call activity
- I/O activity including network & file access

When used in a Kubernetes environment, Tetragon is Kubernetes-aware - that is, it understands Kubernetes identities such as namespaces, pods and so-on - so that security event detection can be configured in relation to individual workloads.



Sicherheitsanwendungen: Falco

Cloud Native Runtime Security.

Want to talk? Join us on the #falco channel in the Kubernetes Slack.

Sicherheitsanwendungen: tracee

Tracee: Runtime Security and Forensics using eBPF

Tracee is a Runtime Security and forensics tool for Linux. It uses Linux **eBPF technology** to trace your system and applications **at runtime**, and analyzes collected events in order to detect **suspicious behavioral patterns**. It is usually delivered as a docker container, but there are other ways you can use it (even create your own customized tracee container).

1.7.13. Quis custodiet ipsos custodes?

Backdoors

- Mittels eBPF lassen sich potente Backdoors im Betriebssystem–Kernel *verstecken*
  - Persistent?
  - In der Netzwerk–Karte?
  - Um eBPF Programme in den Kernel zu laden werden priviligierte Rechte (CAP_SYS_ADMIN) benötigt (in neueren Linux Distributions–Versionen)
    - `/proc/sys/kernel/unprivileged_bpf_disabled`

boopkit

```
================================================================

    ▄▄▄▄▄▄  ▄▄▄▄▄▄  ▄▄▄▄▄▄  ▄▄▄▄▄▄  ▄▄  ▄▄  ▄▄  ▄▄▄▄▄▄▄▄
    ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄
    ▄▄▄▄▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄▄▄▄▄  ▄▄▄▄▄▄  ▄▄  ▄▄
    ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄
    ▄▄▄▄▄▄  ▄▄▄▄▄▄  ▄▄▄▄▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄  ▄▄

    Author: Kris Nóva <kris@nivenly.com> Version 1.4.0

    IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
    LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
    EXEMPLARY, OR CONSEQUENTIAL DAMAGES.

    DO NOT ATTEMPT TO USE THE TOOLS TO VIOLATE THE LAW.
    THE AUTHOR IS NOT RESPONSIBLE FOR ANY ILLEGAL ACTION.
    MISUSE OF THE SOFTWARE, INFORMATION, OR SOURCE CODE
    MAY RESULT IN CRIMINAL CHARGES.

    Use at your own risk.

================================================================

Boopkit.
Linux rootkit and backdoor. Built using eBPF.

Usage:
boopkit [options]

Options:
-h, help            Display help and usage for boopkit.
-i, interface       Interface name. lo, eth0, wlan0, etc
-s, sudo-bypass     Bypass sudo check. Breaks PID obfuscation.
-r, reverse-conn    Attempt a reverse RCE lookup if no payload found.
-q, quiet           Disable output.
-x, reject          Source addresses to reject triggers from.
```

TripeCross

TripleCross

TripleCross is a Linux eBPF rootkit that demonstrates the offensive capabilities of the eBPF technology.

TripleCross is inspired by previous implant designs in this area, notably the works of Jeff Dileo at DEFCON 27[1], Pat Hogan at DEFCON 29[2], Guillaume Fournier and Sylvain Afchain also at DEFCON 29[3], and Kris Nóva's Boopkit[4]. We reuse and extend some of the techniques pioneered by these previous explorations of the offensive capabilities of eBPF technology.

This rootkit was created for my Bachelor's Thesis at UC3M. More details about its design are provided in the thesis document.

BPFdoor

Symbiote

Home > Security > Malware

NEWS ANALYSIS

Hackers using stealthy Linux backdoor Symbiote to steal credentials

Symbiote is deployed as a shared object that can inject itself into existing processes, making it difficult to detect.

Bvp47

Anatomy of suspected top-tier decade-hidden NSA backdoor

Bvp47 of yore said to have used BPF to conceal comms in network traffic

Thomas Claburn in San Francisco

Wed 23 Feb 2022 // 20:23 UTC

19

Pangu Lab has identified what it claims is a sophisticated backdoor that was used by the NSA to subvert highly targeted Linux systems around the world for more than a decade.

The China-based computer-security outfit says it first spotted the backdoor code, or advanced persistent threat (APT), in 2013 when conducting a forensic investigation on a host in "a key domestic department" – presumably a Chinese company or government agency.

To us it seems whoever created the code would compromise or infect a selected Linux system and then install the backdoor on it. This backdoor, which Pangu has now described, would do its best to hide from administrators and users, and covertly communicate over networks with the outside world.

1.7.14. eBPF einschränken

"Unprivilegded eBPF" ausschalten

- Bis 2022 hatten einige (populäre) Linux Distributionen *unpriviligiertes eBPF* aktiviert
  - **Jeder** Benutzer konnte eBPF Code in den Kernel laden!
  - Ubuntu (und andere Linux-Distributionen) haben diese Lücke in Frühjar 2022 geschlossen
  - Prüfen, daß die Linux Kernel-Variable (sysctl) `kernel.unprivileged_bpf_disabled` > `0` ist

KPROBE_OVERRIDE im Kernel

- Ist in der Kernel Konfiguration der Schalter `CONFIG_BPF_KPROBE_OVERRIDE` aktiv (zur Übersetzungszeit des Kernels), so können eBPF Programme die Rückgabewerte von (Kernel-)Funktionen überschreiben
- Diese Konfiguration sollte in Kernel in Produktions-Umgebungen **nicht** gesetzt sein (Kernel "config" prüfen)

KProbes ausschalten

- Durch schreiben des Wertes `0` in die Pseudo-Datei `/sys/kernel/debug/kprobes/enabled` werden die Kernel-Probes (KPROBES) ausgeschaltet
- Achtung: ein Benutzer mit Schreibrechten auf diese Datei (z.B. `root`) kann die KPROBES wieder anschalten

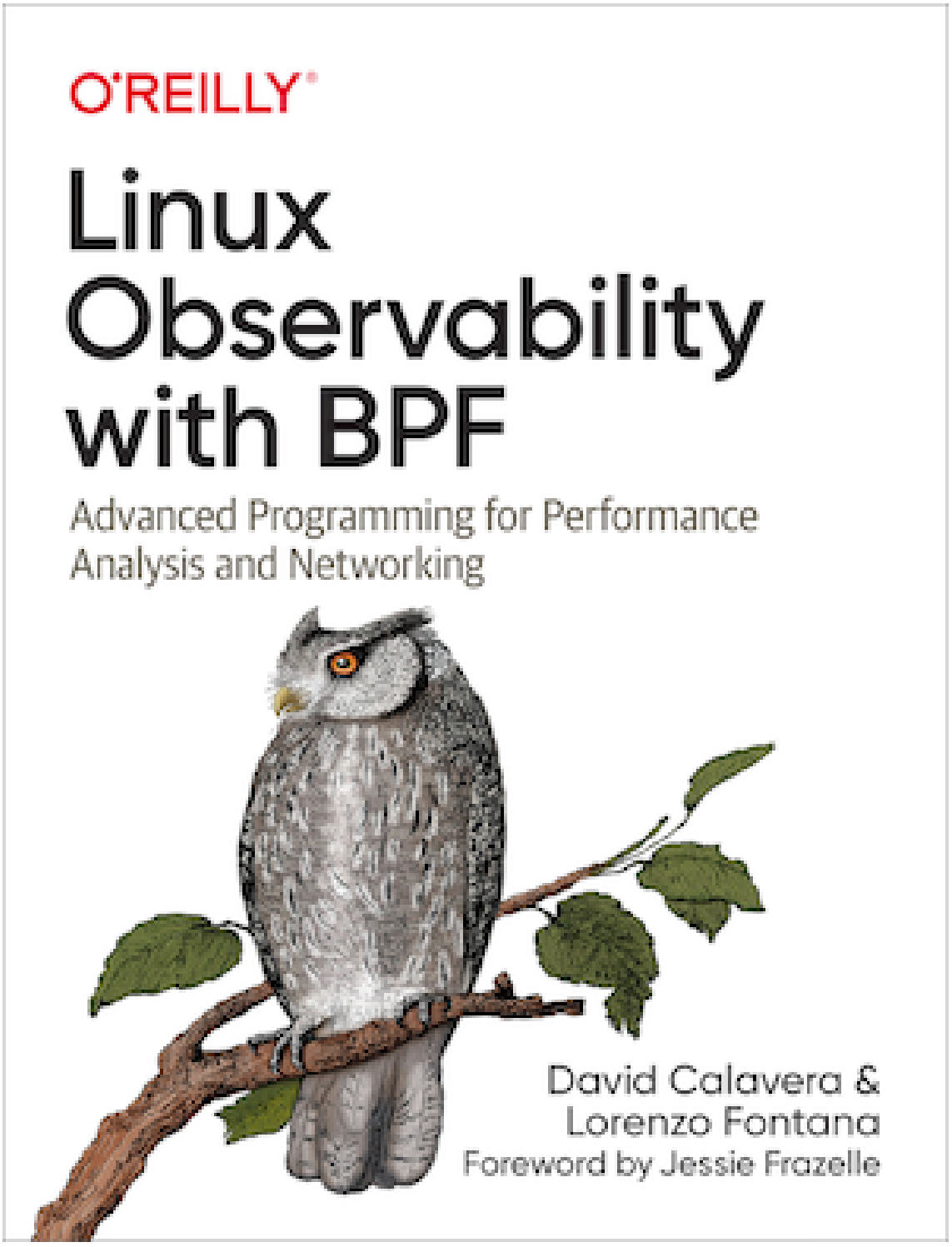
Kernel ohne eBPF

- In sicherheitskritischen Bereichen sollten ggf. Kernel ohne eBPF Funktion (kprobes, XDP, eBPF TC Filter) eingesetzt werden
  - Hierzu muss der Kernel neu übersetzt werden (dies ist oft nicht praktikabel)
  - Es gibt (derzeit, 2023) keinen Kernel Commandline Schalter um eBPF auszuschalten

1.7.15. eBPF Literatur

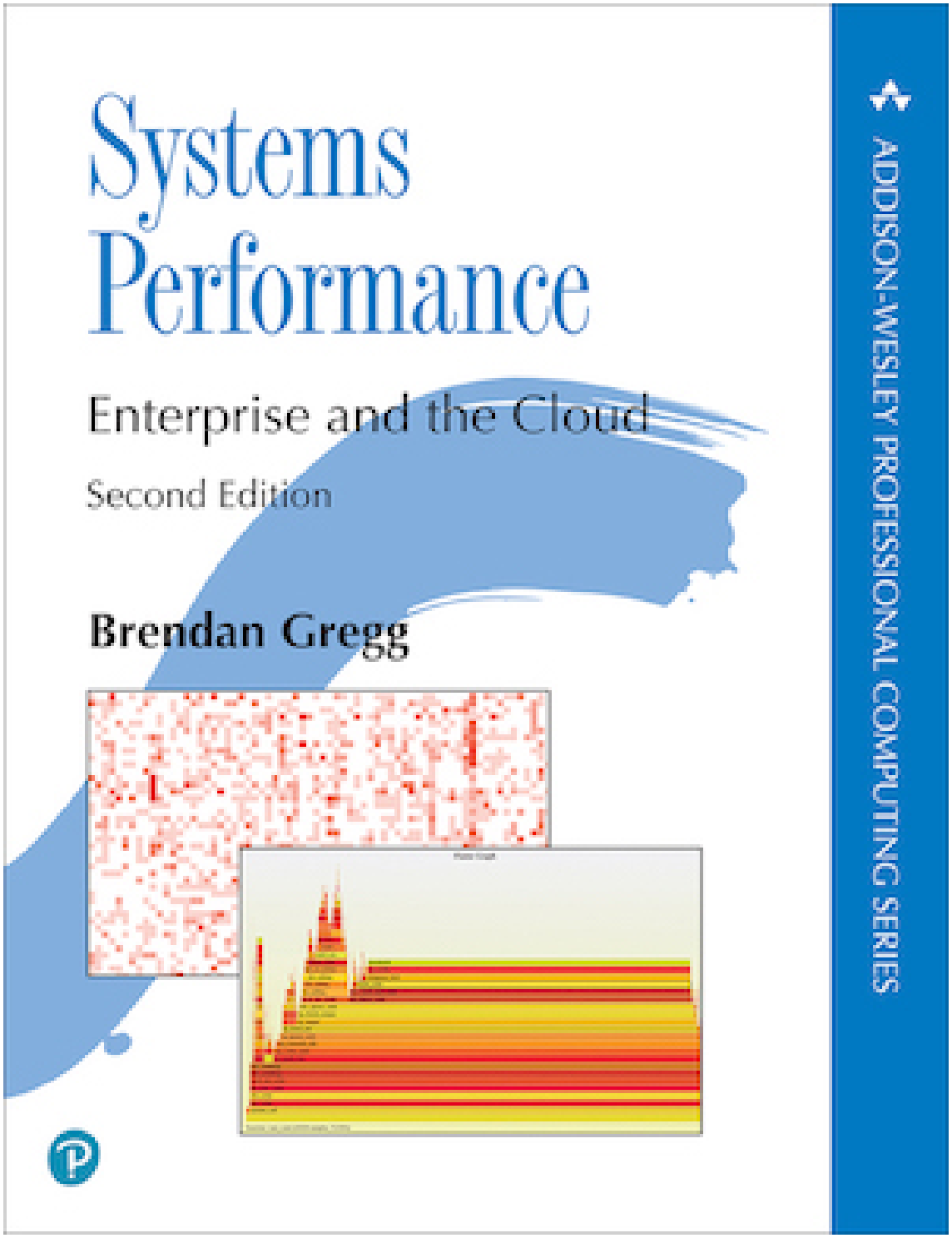
Buch: Linux Observability with BPF

Von David Calavera, Lorenzo Fontana (November 2019)


O'REILLY
Linux Observability with BPF
Advanced Programming for Performance Analysis and Networking
David Calavera &
Lorenzo Fontana
Foreword by Jessie Frazelle

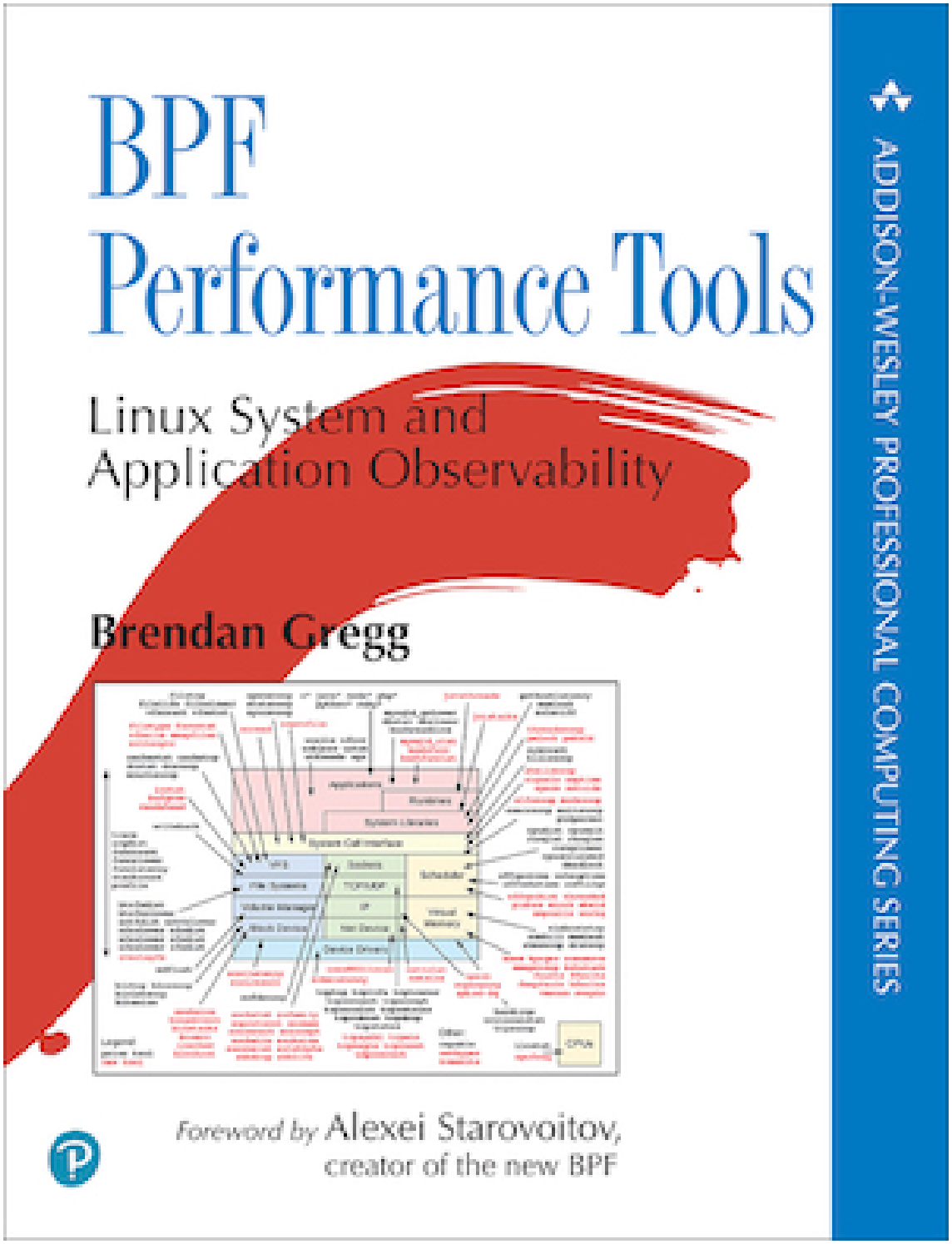
Buch: Systems Performance (2nd ed.)

Von Brendan Gregg (Dezember 2020)



Buch: BPF Performance Tools

Von Brendan Gregg (Dezember 2019)

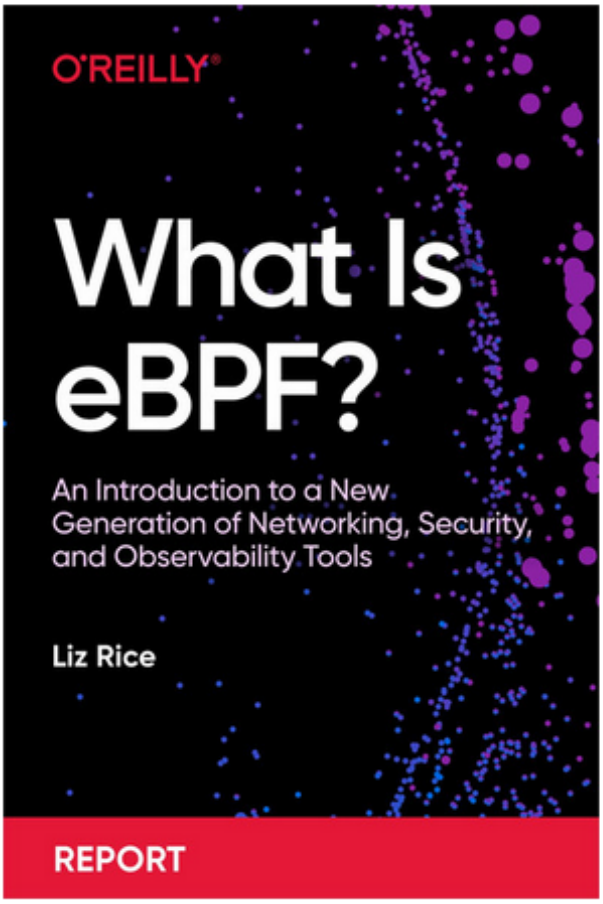


Report: What is eBPF? (O'Reilly)

What Is eBPF?

★★★★★ 1 review

By Liz Rice



TIME TO COMPLETE:
1h 7m

TOPICS:
Linux

PUBLISHED BY:
O'Reilly Media, Inc.

PUBLICATION DATE:
April 2022

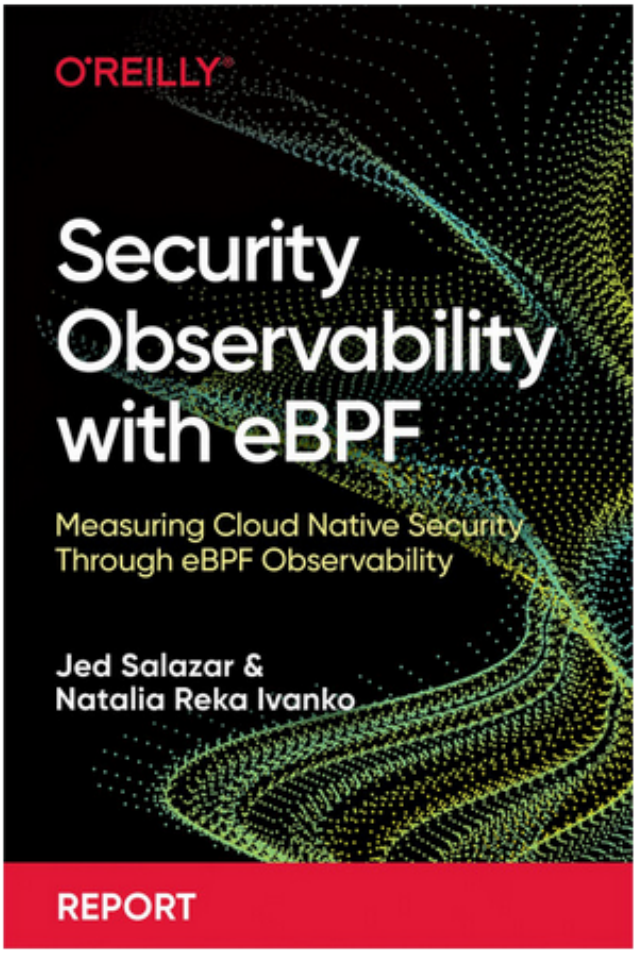
PRINT LENGTH:
52 pages

Report: Security Observability with eBPF (O'Reilly)

Security Observability with eBPF

Write the first review

By Jed Salazar, Natalia Reka Ivanko



TIME TO COMPLETE:
1h 28m

TOPICS:
Kubernetes

PUBLISHED BY:
O'Reilly Media, Inc.

PUBLICATION DATE:
April 2022

PRINT LENGTH:
65 pages

1.7.16. eBPF Links

eBPF

- eBPF support and Linux kernel versions https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md [https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md]
- Awesome BPF https://github.com/zoidbergwill/awesome-ebpf [https://github.com/zoidbergwill/awesome-ebpf]
- Using user-space tracepoints with BPF https://lwn.net/Articles/753601/ [https://lwn.net/Articles/753601/]

- Extending systemd Security Features with eBPF https://kinvolk.io/blog/2021/04/extending-systemd-security-features-with-ebpf/ [https://kinvolk.io/blog/2021/04/extending-systemd-security-features-with-ebpf/]
- Absolute Beginner's Guide to BCC, XDP, and eBPF https://dev.to/satrobit/absolute-beginner-s-guide-to-bcc-xdp-and-ebpf-47oi [https://dev.to/satrobit/absolute-beginner-s-guide-to-bcc-xdp-and-ebpf-47oi]
- Linux Extended BPF (eBPF) Tracing Tools https://www.brendangregg.com/ebpf.html [https://www.brendangregg.com/ebpf.html]
- Performance Implications of Packet Filtering with Linux eBPF https://www.net.in.tum.de/fileadmin/bibtex/publications/papers/ITC30-Packet-Filtering-eBPF-XDP.pdf [https://www.net.in.tum.de/fileadmin/bibtex/publications/papers/ITC30-Packet-Filtering-eBPF-XDP.pdf]
- eBPF for perfomance analysis and networking https://marioskogias.github.io/students/debeule.pdf [https://marioskogias.github.io/students/debeule.pdf]
    - BPF and XDP Reference Guide https://docs.cilium.io/en/v1.10/bpf/ [https://docs.cilium.io/en/v1.10/bpf/]

BCC

- Intro to Kernel and Userspace Tracing Using BCC, Part 1 of 3 https://blogs.oracle.com/linux/post/intro-to-bcc-1 [https://blogs.oracle.com/linux/post/intro-to-bcc-1]

bpftrace

- bpftrace Reference Guide https://github.com/iovisor/bpftrace/blob/master/docs/reference_guide.md [https://github.com/iovisor/bpftrace/blob/master/docs/reference_guide.md]
- Kernel analysis with bpftrace https://lwn.net/Articles/793749/ [https://lwn.net/Articles/793749/]
- The bpftrace One-Liner Tutorial https://github.com/iovisor/bpftrace/blob/master/docs/tutorial_one_liners.md [https://github.com/iovisor/bpftrace/blob/master/docs/tutorial_one_liners.md]
- Full-system dynamic tracing on Linux using eBPF and bpftrace https://www.joyfulbikeshedding.com/blog/2019-01-31-full-system-dynamic-tracing-on-linux-using-ebpf-and-bpftrace.html [https://www.joyfulbikeshedding.com/blog/2019-01-31-full-system-dynamic-tracing-on-linux-using-ebpf-and-bpftrace.html]
- bpftrace Cheat Sheet https://www.brendangregg.com/BPF/bpftrace-cheat-sheet.html [https://www.brendangregg.com/BPF/bpftrace-cheat-sheet.html]

Network Scripts

- udplife https://github.com/brendangregg/bpf-perf-tools-book/blob/master/exercinses/Ch10_Networking/udplife.bt [https://github.com/brendangregg/bpf-perf-tools-book/blob/master/exercinses/Ch10_Networking/udplife.bt]
- How SKBs work http://vger.kernel.org/}davem/skb_data.html [http://vger.kernel.org/<sub>davem/skb_data.html]

eBPF Prometheus exporter

- eBPF exporter https://blog.cloudflare.com/introducing-ebpf_exporter/ [https://blog.cloudflare.com/introducing-ebpf_exporter/]

eXpress Data Path (XDP)

- Introduction to: XDP and BPF building blocks https://people.netfilter.org/hawk/presentations/ebplane2019/xdp-bpf-building-blocks.pdf [https://people.netfilter.org/hawk/presentations/ebplane2019/xdp-bpf-building-blocks.pdf]
- A practical introduction to XDP https://www.linuxplumbersconf.org/event/2/contributions/71/attachments/17/9/presentation-lpc2018-xdp-tutorial.pdf [https://www.linuxplumbersconf.org/event/2/contributions/71/attachments/17/9/presentation-lpc2018-xdp-tutorial.pdf]
- eBPF/XDP https://www.slideshare.net/Netronome/ebpfxdp-sigcomm-2018 [https://www.slideshare.net/Netronome/ebpfxdp-sigcomm-2018]
- XDP Packet filter and UDP https://fly.io/blog/bpf-xdp-packet-filters-and-udp/ [https://fly.io/blog/bpf-xdp-packet-filters-and-udp/]
- XDP Firewall https://github.com/gamemann/XDP-Firewall [https://github.com/gamemann/XDP-Firewall]

eXpress Data Path (XDP)

- How to filter packets super fast: XDP & eBPF! https://jvns.ca/blog/2017/04/07/xdp-bpf-tutorial/ [https://jvns.ca/blog/2017/04/07/xdp-bpf-tutorial/]
- Load XDP programs using the ip (iproute2) command https://medium.com/@fntlnz/load-xdp-programs-using-the-ip-iproute2-command-502043898263 [https://medium.com/@fntlnz/load-xdp-programs-using-the-ip-iproute2-command-502043898263]
- L4Drop: XDP DDoS Mitigations https://blog.cloudflare.com/l4drop-xdp-ebpf-based-ddos-mitigations/ [https://blog.cloudflare.com/l4drop-xdp-ebpf-based-ddos-mitigations/]
- How to drop a packet in Linux in more ways than one https://codilime.com/blog/how-to-drop-a-packet-in-linux-in-more-ways-than-one/ [https://codilime.com/blog/how-to-drop-a-packet-in-linux-in-more-ways-than-one/]
- eBPFsnitch https://github.com/harporoeder/ebpfsnitch [https://github.com/harporoeder/ebpfsnitch]
- XDP minimal example https://ruderich.org/simon/notes/xdp-minimal-example [https://ruderich.org/simon/notes/xdp-minimal-example]
- Why is the kernel community replacing iptables with BPF? https://cilium.io/blog/2018/04/17/why-is-the-kernel-community-replacing-iptables [https://cilium.io/blog/2018/04/17/why-is-the-kernel-community-replacing-iptables]

BPF Backdoor

- Linux eBPF backdoor over TCP https://github.com/kris-nova/boopkit [https://github.com/kris-nova/boopkit]
- A Linux eBPF rootkit with a backdoor, C2, library injection, execution hijacking, persistence and stealth capabilities https://github.com/h3xduck/TripleCross [https://github.com/h3xduck/TripleCross]

- Offensive BPF: Using bpftrace to host backdoors https://embracethered.com/blog/posts/2021/offensive-bpf-bpftrace-message-based/ [https://embracethered.com/blog/posts/2021/offensive-bpf-bpftrace-message-based/]
- Offensive BPF: Malicious bpftrace https://embracethered.com/blog/posts/2021/offensive-bpf-bpftrace/ [https://embracethered.com/blog/posts/2021/offensive-bpf-bpftrace/]

BPF Malware/Angriffe

- BPFDoor: Chinese tool almost undetected for FIVE years is second BPF-based attack uncovered this year https://linuxsecurity.com/news/hackscracks/bpfdoor-chinese-tool-almost-undetected-for-five-years-is-second-bpf-based-attack-uncovered-this-year [https://linuxsecurity.com/news/hackscracks/bpfdoor-chinese-tool-almost-undetected-for-five-years-is-second-bpf-based-attack-uncovered-this-year]
- Anatomy of suspected top-tier decade-hidden NSA backdoor https://www.theregister.com/2022/02/23/chinese_nsa_linux/ [https://www.theregister.com/2022/02/23/chinese_nsa_linux/]

BPF Malware/Angriffe

- Hackers using stealthy Linux backdoor Symbiote to steal credentials https://www.csoonline.com/article/3663510/hackers-using-stealthy-linux-backdoor-symbiote-to-steal-credentials.html [https://www.csoonline.com/article/3663510/hackers-using-stealthy-linux-backdoor-symbiote-to-steal-credentials.html]
- Bvp47 – Top-tier Backdoor of US NSA Equation Group https://www.pangulab.cn/files/The_Bvp47_a_top-tier_backdoor_of_us_nsa_equation_group.en.pdf [https://www.pangulab.cn/files/The_Bvp47_a_top-tier_backdoor_of_us_nsa_equation_group.en.pdf]

eBPF Sicherheit

- Mapping It Out: Analyzing the Security of eBPF Maps https://www.crowdstrike.com/blog/analyzing-the-security-of-ebpf-maps/ [https://www.crowdstrike.com/blog/analyzing-the-security-of-ebpf-maps/]
- With Friends like eBPF, who needs enemies ? https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-With-Friends-Like-EBPF-Who-Needs-Enemies.pdf [https://i.blackhat.com/USA21/Wednesday-Handouts/us-21-With-Friends-Like-EBPF-Who-Needs-Enemies.pdf]
- Kernel Pwning with eBPF: a Love Story https://www.graplsecurity.com/post/kernel-pwning-with-ebpf-a-love-story [https://www.graplsecurity.com/post/kernel-pwning-with-ebpf-a-love-story]

eBPF Sicherheit

- Ubuntu is releasing updated kernels that disable unprivileged eBPF by default https://wiki.ubuntu.com/SecurityTeam/KnowledgeBase/BHI [https://wiki.ubuntu.com/SecurityTeam/KnowledgeBase/BHI]
- Toward signed BPF programs https://lwn.net/Articles/853489/ [https://lwn.net/Articles/853489/]
- CONFIG_BPF_KPROBE_OVERRIDE: Enable BPF programs to override a kprobed function https://cateee.net/lkddb/web-lkddb/BPF_KPROBE_OVERRIDE.html [https://cateee.net/lkddb/web-

lkddb/BPF_KPROBE_OVERRIDE.html]

- The kprobes debugfs interface https://www.kernel.org/doc/Documentation/kprobes.txt [https://www.kernel.org/doc/Documentation/kprobes.txt]

eBPF Sicherheitsprobleme

- CVE-2021-33624 kernel: Linux kernel BPF protection against speculative execution attacks can be bypassed to read arbitrary kernel memory

eBPF Sicherheitswerkzeuge

- A flow-based IDS using Machine Learning in eBPF https://arxiv.org/abs/2102.09980 [https://arxiv.org/abs/2102.09980]
- ebpfkit-monitor is a tool that detects and protects against eBPF powered rootkits https://github.com/Gui774ume/ebpfkit-monitor [https://github.com/Gui774ume/ebpfkit-monitor]
- MAC and Audit policy using eBPF (KRSI) https://lwn.net/Articles/813057/ [https://lwn.net/Articles/813057/]
- KRSI — the other BPF security module https://lwn.net/Articles/808048/ [https://lwn.net/Articles/808048/]

eBPF Audit-Tools

- bpfcontain-rs – https://github.com/willfindlay/bpfcontain-rs/ [https://github.com/willfindlay/bpfcontain-rs/]
- Tracee – Linux Runtime Security and Forensics using eBPF https://github.com/aquasecurity/tracee [https://github.com/aquasecurity/tracee]
- falco – Cloud Native Runtime Security https://github.com/falcosecurity/falco [https://github.com/falcosecurity/falco]
- Cilium Tetragon – eBPF-based Security Observability and Runtime Enforcement https://github.com/cilium/tetragon [https://github.com/cilium/tetragon]

1.7.17. eBPF Praxis

Discover eBPF programs with `bpftool`

- Installing `bpftool`:

```
% dnf install bpftool
```

- Which `eBPF` programms are loaded into the Linux Kernel?

```
% bpftool prog list
306: cgroup_device  tag ca8e50a3c7fb034b  gpl
        loaded_at 2023-02-09T21:16:16+0000  uid 0
        xlated 496B  jited 307B  memlock 4096B
        pids systemd(1)
307: cgroup_skb  tag 6deef7357e7b4530  gpl
        loaded_at 2023-02-09T21:16:16+0000  uid 0
```

```
        xlated 64B  jited 54B  memlock 4096B
        pids systemd(1)
308: cgroup_skb  tag 6deef7357e7b4530  gpl
        loaded_at 2023-02-09T21:16:16+0000  uid 0
[...]
```

- What are these? See `cgroup_skb`
  - https://github.com/systemd/systemd/blob/main/src/core/cgroup.c [https://github.com/systemd/systemd/blob/main/src/core/cgroup.c]
  - https://aya-rs.dev/book/programs/cgroup-skb/ [https://aya-rs.dev/book/programs/cgroup-skb/]
- Are *eBPF* programm statistics enabled?

```
% sysctl kernel.bpf_stats_enabled
kernel.bpf_stats_enabled = 0
```

- Let's enable *eBPF* statistics

```
% sysctl -w kernel.bpf_stats_enabled=1
```

- Dump disassembled *eBPF* source code

```
% # bpftool prog dump xlated id 317
   0: (bf) r6 = r1
   1: (69) r7 = *(u16 *)(r6 +176)
   2: (b4) w8 = 0
   3: (44) w8 |= 2
   4: (b7) r0 = 1
   5: (55) if r8 != 0x2 goto pc+1
   6: (b7) r0 = 0
   7: (95) exit
```

- See the *eBPF* JIT compiler generated machine code disassembly

```
% bpftool prog dump jited tag tag 6deef7357e7b4530
317: cgroup_skb  tag 6deef7357e7b4530  gpl
[6/723]
0xffffffffc0af90d0:
   0:   nopl   0x0(%rax,%rax,1)
   5:   xchg   %ax,%ax
   7:   push   %rbp
   8:   mov    %rsp,%rbp
   b:   push   %rbx
   c:   push   %r13
   e:   push   %r14
  10:   mov    %rdi,%rbx
  13:   movzwq 0xb0(%rbx),%r13
[...]
```

bpttrace tools and on-liners

- Install the BIND 9 DNS Server

```
% dnf install bind
% systemctl enable --now named
```

- Install the bpftrace tools

```
% dnf install bpftrace
```

- Switch the operating system to use the BIND 9 DNS resolver

```
% echo "nameserver 127.0.0.1" > /etc/resolv.conf
```

- Example 1: Start `gethostlatency.bt` (found in `/usr/share/bpftrace/tools`) in one terminal, execute some other network tools that require DNS name resolution (`ping`, `dnf`, `traceroute` etc) to see the DNS name resolution latency
- Example 2: Count UDP send/receives by on-CPU PID and process name (Script is an excerpt From "BPF Performance Tools" by Brendan Gregg)
  - Execute some DNS queries against the BIND 9 resolver
  - Exec the bpftrace script with CTRL+C to see the report

```
% bpftrace -e "k:udp*_sendmsg,k:udp*_recvmsg { @[func, pid, comm] = count(); }"
```

- Example 3: Show packets received over UDP by the `named` process as a histogram (execute some queries, then terminate the script with CTRL+C)

```
% bpftrace -e "kr:udp_recvmsg /pid == $(pgrep named)/ { @recv_bytes = hist(retval); }"
```

- Example 4: Histogram of UDP packets send by the BIND 9 process (execute some queries, then terminate the script with CTRL+C)

```
% bpftrace -e "kprobe:udp_sendmsg /pid == $(pgrep named)/ { @size = hist(arg2); }"
```

- Example 5: BIND 9 tracing – print whenever `rndc status` is executed

```
% bpftrace -e 'uprobe:named:named_server_status { print("rndc status executed") }'
```

- Example 6: How long does it take (in nanoseconds) to flush the cache with `rndc flush`? Do some queries against the BIND 9 DNS resolver to fill the cache. Then call `rndc flush` to flush the cache. The printed value should get higher the more cache entries need to be cleaned.

```
% bpftrace -e "uprobe:/usr/sbin/named:named_server_flushcache / pid == $(pgrep named)
/{ @start[tid] = nsecs; }
    uretprobe:/usr/sbin/named:named_server_flushcache /@start[tid]/ { print(nsecs -
@start[tid]); delete(@start[tid]); }"
```

- Trace the cache related functions BIND 9 executes. Send some queries to the BIND 9 DNS resolver, then use `rndc flush`, `rndc flushname <domain-name>` and `rndc flushtree <name>` and see which functions are executed inside BIND 9

```
% bpftrace -e 'uprobe:/usr/lib64/libdns-9.16.23-RH.so:dns_cache* { print(func) }'
```

Instrumenting BIND 9

- Configure BIND 9 to forward all request for the domain `isc.org` to Quad9 (9.9.9.9). In the file `/etc/named.conf` add

```
zone "isc.org" {
  type forward;
  forwarders { 9.9.9.9; };
};
```

- Check the configuration with `named-checkconf` and restart the BIND 9 DNS server with `systemctl restart named`
- Install `bpftrace`

```
dnf install bpftrace
```

- Save the following source into the file `forward-trace.bt`

```
#!/usr/bin/bpftrace

struct dns_name {
        unsigned int   magic;
        unsigned char *ndata;
        unsigned int   length;
        unsigned int   labels;
        unsigned int   attributes;
        unsigned char *offsets;
//      isc_buffer_t  *buffer;
//      ISC_LINK(dns_name_t) link;
//      ISC_LIST(dns_rdataset_t) list;
};

BEGIN
{
  print("Waiting for forward decision...\n");
}
uprobe:/usr/lib64/libdns-9.16.23-RH.so:dns_fwdtable_find
{
  @dns_name[tid] = ((struct dns_name *)arg1)->ndata
}

uretprobe:/usr/lib64/libdns-9.16.23-RH.so:dns_fwdtable_find
{
 if (retval == 0) {
    printf("Forwarded domain name: %s\n", str(@dns_name[tid]));
 }
 delete(@dns_name[tid]);
}
```

- Make the file executeable

```
% chmod +x forward-trace.bt
```

- Execute the `bpftrace` script

```
% ./forward-trace.bt
```

- Login to the lab machine from a different terminal (or use `tmux`). Flush the cache of the BIND 9 resolver with `rndc flush`, then execute a DNS name resolution for `isc.org` and for other domains. See that the `bpftrace` script reports the forward decisions in BIND 9

```
% dig @localhost isc.org
```

- Our eBPF program in the `bpftool` listing

```
bpftool prog list | tail
        xlated 64B  jited 54B  memlock 4096B
        pids systemd(1)
383: kprobe  name uretprobe__dns_  tag 30ca488d1d146cd7  gpl run_time_ns 297757 run_cnt
115
        loaded_at 2023-02-09T21:46:24+0000  uid 0
        xlated 536B  jited 314B  memlock 4096B  map_ids 25,26
        pids forward-trace.b(34273)
384: kprobe  name uprobe__dns_fwd  tag c30da15c9fd45317  gpl run_time_ns 116583 run_cnt
115
        loaded_at 2023-02-09T21:46:24+0000  uid 0
        xlated 176B  jited 105B  memlock 4096B  map_ids 25
        pids forward-trace.b(34273)
```

- Try to disassemble the *eBPF* programm (eBPF code and JIT)

Using XDP/eBPF to drop all UDP except DNS

- Install the Kernel headers for the running Linux Kernel. These headers are needed to get the function names and entry points into the Linux Kernel for the eBPF probes

```
  % dnf install kernel-headers-$(uname -r)
```

- Create a text file with the name `drop-non-dns-udp.c` and enter the following C-Code. This eBPF program will ...
  - ... be executed for every incoming network packet
  - ... prints the text *got a packet* whenever an IP packet is received
  - ... extracts the IP- and UDP-header from the packet
  - ... if the packet neither has the UDP source port of 53 (DNS) nor a destination port of 53 the return code of `XDP_DROP` will be returned, which will discard the packet. A message is printed to inform about the dropped packet
  - ... all other network packets will be given with the return value of `XDP_PASS` into the Linux TCP/IP stack for further processing
  - This is an example XDP program. Production quality eBPF/XDP programms should use the eBPF map structures to exchange information with the User-Space program instead of using the `bpf_trace_printk` function.

```
#define KBUILD_MODNAME "filter"
```

```
#include <linux/bpf.h>
#include <linux/if_ether.h>
#include <linux/ip.h>
#include <linux/in.h>
#include <linux/udp.h>

int udpfilter(struct xdp_md *ctx) {
  bpf_trace_printk("got a packet\n");
  void *data = (void *)(long)ctx->data;
  void *data_end = (void *)(long)ctx->data_end;
  struct ethhdr *eth = data;
  if ((void*)eth + sizeof(*eth) <= data_end) {
    struct iphdr *ip = data + sizeof(*eth);
    if ((void*)ip + sizeof(*ip) <= data_end) {
      if (ip->protocol == IPPROTO_UDP) {
        struct udphdr *udp = (void*)ip + sizeof(*ip);
        if ((void*)udp + sizeof(*udp) <= data_end) {
          if ((udp->dest != ntohs(53)) && (udp->source != ntohs(53))) {
            if (udp->dest != udp->source) {
              bpf_trace_printk("drop udp src/dest port %d/%d\n", ntohs(udp->source),
ntohs(udp->dest));
              return XDP_DROP;
            }
          }
        }
      }
    }
  }
  return XDP_PASS;
}
```

- Also create a eBPF loader program (written in Python) in the file `drop-non-dns-udp.py`.
    - This loader will compile the C-Program above into eBPF byte code, load the program into the Linux Kernel and will detach the eBPF program with the loopback network interface (`lo`)
    - There will be a few warnings issued during compilation. This is not a problem for this exercise and we can ignore these for this lab session
    - The `virtio` network driver of the virtual machine environment doas not allow eBPF programs on the regular network interfaces (`eth0` and `eth1`), therefore we test this function with the lookback interface.

```
#!/usr/bin/env python3

from bcc import BPF
import time

device = "lo"
b = BPF(src_file="drop-non-dns-udp.c")
fn = b.load_func("udpfilter", BPF.XDP)
b.attach_xdp(device, fn, 0)

try:
  b.trace_print()
```

```
except KeyboardInterrupt:
  pass

b.remove_xdp(device, 0)
```

- Make the loader executable

```
% chmod +x drop-non-dns-udp.py
```

- Execute the loader program (the warnings can be ingnored)

```
% ./drop-non-dns-udp.py
```

- Create a new connection to the lab machines (new browser tab, another SSH connection or use `tmux`)
- Create DNS queries towards the running BIND 9 DNS resolver over the loopback interface. These queries shoul **not** be blocked:

```
% dig @localhost isc.org
```

- Send UDP packets (DNS or other protocol) towards a port other than 53, the packets will be discarded **before** entering the regular Linux Kernel TCP/IP stack:

```
% dig -p 5353 @localhost isc.org
```

1.8. 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, Chrome, systemd-nspawn, LXC/LXD, Firejail etc. benutzen die Namespaces im Linux-Kernel
- Verfügbare Namespaces in Linux

Namespace	Konstante	Isolation
Cgroup	CLONE_NEWCGROUP	Cgroup root Verzeichnis (Ressourcen wie CPU/RAM)
IPC	CLONE_NEWIPC	System V IPC, POSIX message queues
Network	CLONE_NEWNET	Network devices, stacks, ports, Firewall etc.
Mount	CLONE_NEWNS	Mount points (Dateisysteme)

Namespace	Konstante	Isolation
PID	CLONE_NEWPID	Process IDs
User	CLONE_NEWUSER	Benutzer und Gruppen IDs
UTS	CLONE_NEWUTS	Hostname und NIS Domain Name

1.8.1. 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

```
dnf install gcc make
```

- 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);
  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

CHAPTER 2. TAG 2

2.1. NAMESPACES (TEIL 2)

2.1.1. 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 ein IP-Adresse auf **veth1** im Namespace

```
# ip netns exec netns1 ip addr add 10.1.1.1/24 dev veth1
# ip netns exec netns1 ip link set dev dev veth1 up
# 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 ip route show
ip netns exec netns1 iptables -L
```

- Namespace wieder löschen

```
ip netns delete netns1
```

2.1.2. 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
```

2.1.3. 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 Rocky-Linux 9
  - Andere Linux-Distributionen wie Ubuntu, Suse, Arch-Linux, Alpine-Linux oder 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)
- Systemd-Container Werkzeuge installieren

```
 dnf install systemd-container
```

- Wir erstellen ein Verzeichnis für den neuen Container

```
mkdir -p /srv/container/rocky-linux8
```

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

```
rpm --root /srv/container/rocky-linux8 --initdb
```

- Das Basis-Paket für Rocky-Linux laden

```
curl -O https://ftp.plusline.net/rockylinux/8/BaseOS/x86_64/os/Packages/r/rocky-
release-8.10-1.9.el8.noarch.rpm
curl -O https://ftp.plusline.net/rockylinux/8/BaseOS/x86_64/os/Packages/r/rocky-repos-
8.10-1.9.el8.noarch.rpm
rpm --nodeps --root /srv/container/rocky-linux8 -ivh rocky-repos-8.10-
1.9.el8.noarch.rpm rocky-release-8.10-1.9.el8.noarch.rpm
```

- Das Rocky-Linux Basis-System installieren

```
dnf -y --nogpg --releasever=8 --installroot=/srv/container/rocky-linux8 \
  install systemd passwd dnf procps-ng iproute tmux rocky-gpg-keys
echo 8 > /srv/container/rocky-linux8/etc/dnf/vars/releasever
```

- Eine Shell im Container starten

```
systemd-nspawn -D  /srv/container/rocky-linux8
```

- Root-Passwort setzen und neuen Benutzer anlegen

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

- RPM Datenbank im Container neu aufbauen

```
-bash-4.2$ rpmdb --rebuilddb
```

- Shell im Container verlassen

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

- Auf dem Rocky-Linux 9 System Port 80 (http) in der Firewall öffnen

```
firewall-cmd --add-port=80/tcp --permanent
firewall-cmd --reload
```

- Container booten

```
systemd-nspawn -bD  /srv/container/rocky-linux8
```

- 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 Browser unter

http://selinuxXXX.linux-sicherheit.org [http://selinuxXXX.linux-sicherheit.org] erreichbar sein.

- Container stoppen (im Container eingeben)

```
conainter# poweroff
```

- Container mit privatem Netzwerk-Namespace starten:

```
systemd-nspawn -bD  /srv/container/rocky-linux8 --private-network
```

Container mit eigener (virtuellen) Netzwerkkarte (neue MAC-Adresse). `eth0` ist der Name der physischen Netzwerkschnittstelle des Container-Hosts.

```
systemd-nspawn --network-macvlan=eth0 -M rocky01 \
  -bD /srv/container/rocky-linux8
container# ip links
container# dhclient mv-eth0
container# ip address show
container# ping 1.1.1.1
```

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

```
machinectl list
machinectl status rocky01
machinectl terminate rocky01
machinectl kill rocky01
```

- 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
```

2.1.4. 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

```
  dnf install firejail
```

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

```
# dnf install links
```

- Einen im Linux System existierenden unprivilegierten Benutzer (z.B. `user, nutzer` 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 600 .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 - nutzer
$ firejail links https://notes.defaultroutes.de
```

- Die Webseite des Trainings 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`, speichere die Trainings-Webseite im Verzeichnis `Downloads`. 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
```

2.2. LINUX CGROUPS (CONTROLL–GROUPS)

- Laste–Erzeugungs–Tool `stress-ng` installieren

```
dnf install stress-ng
```

- CGroups Informationen anzeigen

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

2.2.1. CGroups manuell

- Stress erzeugen

```
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 +io +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

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

2.2.2. Systemresourcen beschränken mit systemd-run

```
systemd-run stress -c 3
systemd-run stress -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
```

2.2.3. systemctl accounting anschalten

```
    $EDITOR /etc/systemd/system.conf.d/override-accounting.conf
    -----
```

```
    DefaultCPUAccounting=yes
    DefaultIOAccounting=yes
    DefaultBlockIOAccounting=yes
    DefaultMemoryAccounting=yes
    DefaultTasksAccounting=yes
```

2.2.4. systemd

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

2.3. SYSTEMD SICHERHEIT

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

2.3.1. Einfache Direktiven

- Units unter anderer uid/gid laufen lassen
- Zugriff auf Verzeichnisse beschränken
- Prozesslimits setzen

```
$EDITOR /etc/systemd/system/simplehttp.service

[Unit]
Description=HTTP Server

[Service]
Type=simple
Restart=on-failure

User=karl
Group=users

WorkingDirectory=/usr/share/doc
ReadOnlyDirectories=/var
InaccessibleDirectories=/home /etc/ssh
LimitNPROC=1  #darf nicht forken
LimitFSIZE=0  #darf keine Files schreiben

ExecStart=/bin/python3 -m http.server 8000
```

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

- System-Härtung von Systemd-Units unter RedHat EL 8 (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
```

```
RestrictSUIDSGID=yes
```

- Maximaler Speicherverbrauch dieser Anwendung, muss ggf. 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

```
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)**

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

2.3.3. 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

2.3.4. 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
```

2.3.5. Aufgabe:

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

2.4. 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

2.4.1. Dokumentation

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

2.4.2. 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).

2.4.3. Benutzung von journalctl

- Alle Journalmeldungen anzeigen

```
journalclt
```

- 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
```

2.4.4. 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 9Default: 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=436e17-ca6a3d-4310a2-345c98/1d48b1-35a4e900
   PASS:
/run/log/journal/775cf97f988e76f6b34b3904c5817b52/system@2526700792cb4433a4e10ad6e6dedf
cf-000000000000d6db-000622e580c48a69.journal
   3e86c0: Data object references invalid entry at 1147f70
   File corruption detected at
/var/log/journal/a7f85404d5a54e2392d6a92fa98be15a/system.journal:1147e60 (of 25165824
bytes, 72%).
   FAIL: /var/log/journal/a7f85404d5a54e2392d6a92fa98be15a/system.journal (Bad message)
```

- 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]

2.4.5. Journal Red Hat EL 8 Updates

- 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

2.4.6. 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.
```

2.5. 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

2.5.1. 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

```
# dnf install systemd-journal-remote
```

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

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

- In der Firewall Port 19531 freischalten

```
# firewall-cmd --add-port=19531/tcp --permanent
# firewall-cmd --reload
```

- 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

2.5.2. 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

```
# dnf 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
```

- Der Dienst empfängt die Journal–Daten über Port 19532/tcp. Dieser Port wird in der Firewall freigeschalten

```
# firewall-cmd --zone=public --add-port=19532/tcp --permanent
# firewall-cmd --reload
```

- 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
```

- SELinux verhindert ggf. auf dem Sender den Zugriff auf Port 19532

```
# ausearch -ts recent -i
[...]
type=PROCTITLE msg=audit(03/03/2020 15:52:32.957:139) :
proctitle=/usr/lib/systemd/systemd-journal-upload --save-state
type=SYSCALL msg=audit(03/03/2020 15:52:32.957:139) : arch=x86_64 syscall=connect
success=no exit=EACCES(Permission denied) a0=0x7 a1=0x7ffeedeacea0 a2=0x10
a3=0x15dad322800000 items=0 ppid=1 pid=3403 auid=unset uid=systemd-journal-upload
gid=systemd-journal-upload euid=systemd-journal-upload suid=systemd-journal-upload
fsuid=systemd-journal-upload egid=systemd-journal-upload sgid=systemd-journal-upload
fsgid=systemd-journal-upload tty=(none) ses=unset comm=systemd-journal
exe=/usr/lib/systemd/systemd-journal-upload subj=system_u:system_r:init_t:s0 key=(null)
type=AVC msg=audit(03/03/2020 15:52:32.957:139) : avc:  denied  { name_connect } for
pid=3403 comm=systemd-journal dest=19532 scontext=system_u:system_r:init_t:s0
tcontext=system_u:object_r:unreserved_port_t:s0 tclass=tcp_socket permissive=0
```

- SELinux verhindert das beim Emfänger die Journal-Datei angelegt wird

```
# ausearch -ts recent -i
[...]
type=SYSCALL msg=audit(03/03/2020 15:20:44.992:141) : arch=x86_64 syscall=openat
success=no exit=EACCES(Permission denied) a0=0xffffff9c a1=0x557b24ce8ad0
a2=O_RDWR|O_CREAT|O_NONBLOCK|O_CLOEXEC a3=0x1a0 items=0 ppid=1 pid=27829 auid=unset
uid=systemd-journal-remote gid=systemd-journal-remote euid=systemd-journal-remote
suid=systemd-journal-remote fsuid=systemd-journal-remote egid=systemd-journal-remote
sgid=systemd-journal-remote fsgid=systemd-journal-remote tty=(none) ses=unset
```

```
comm=systemd-journal exe=/usr/lib/systemd/systemd-journal-remote
subj=system_u:system_r:init_t:s0 key=(null)
type=AVC msg=audit(03/03/2020 15:20:44.992:141) : avc:  denied  { create } for
pid=27829 comm=systemd-journal name=remote-192.168.1.154.journal
scontext=system_u:system_r:init_t:s0 tcontext=system_u:object_r:var_log_t:s0
tclass=file permissive=0
```

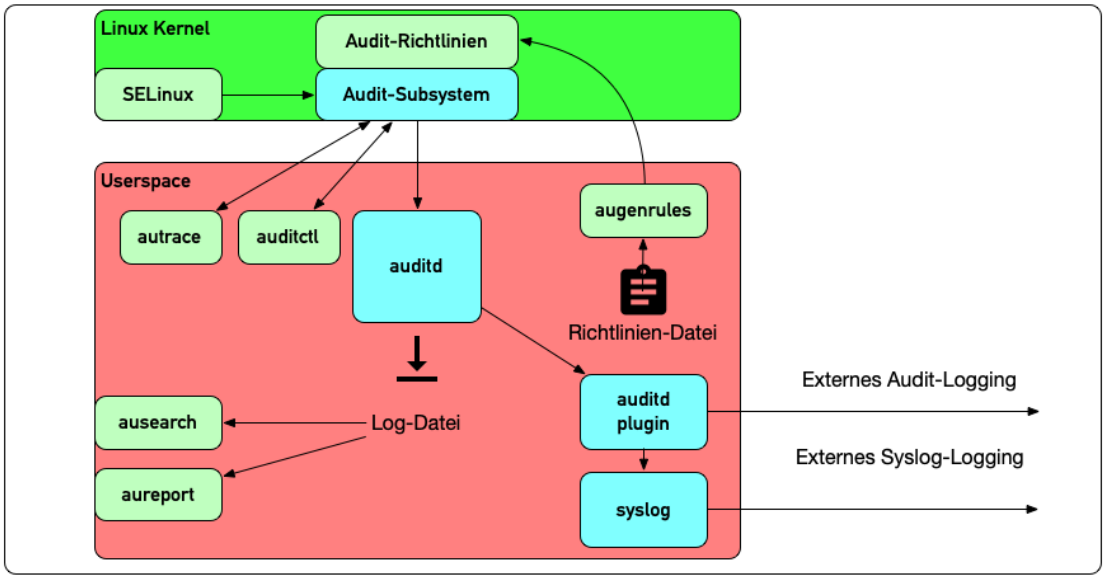
- Lösung: die SELinux Policy anpassen (siehe Kapitel zu SELinux)

2.6. AUDIT SUBSYSTEM

2.6.1. Das Linux Audit Subsystem

- Das Linux Audit Subsystem erlaubt es dem Betreiber eines Linux Systems, ein fein-granulares Audit-Logging für System-Ereignisse festzulegen
  - Ereignisse von Linux-Security-Modulen (LSM) wie AppArmor oder SELinux
  - Ereignisse von sicherheitsrelevanten Anwendugen (SSH, Login-Programm)
  - Ereignisse welche von beliebigen Anwendungen ausgelöst werden und in der Audit-Subsystem Richtlinien-Konfiguration hinterlegt wurden (z.B. Systemcalls, Dateizugriffe, Netzwerk-Aktivitäten)

2.6.2. Audit Subsystem Übersicht



2.6.3. Linux Audit Subsystem Konfiguration

Konfiguration des Audit-Daemon

- Die Konfigurationsdatei des Audit-Daemons unter `/etc/audit/auditd.conf` wurde vom

Audit–Subsystem Projekt in den letzten 20 Jahren nicht aktualisiert und sollte daher vor der Inbetriebnahme des Admit–Subsystems (oder SELinux) auf sinnvolle Werte überprüft werden

Lokal vs. Remote Log Quelle

- Der Audit–Daemon kann Log–Events vom Linux–Kernel oder über das Netzwerk von einem anderen Linux System per Remote–Audit Logging empfangen (Remote Audit Logging ist ein anderes Protokoll als syslog)

```
local_events = yes # "no" to receive remote log-data only
```

- Die Standardeinstellung ist `yes`

Audit–Logdatei

- In der Standardkonfiguration werden die Log–Daten in die Datei `/var/log/audit/audit.log` geschrieben. Für zentrale Audit–Logserver ist es sinnvoll wenn diese Datei auf einem separaten Dateisystem liegt

```
write_logs = yes       # set to "no" for only remote logging
log_file = /var/log/audit/audit.log
```

Gruppenrechte und Format der Audit–Logdatei

- Sollen die Log–Informationen von anderen Benutzer als dem Superuser `root` gelesen werden, so sollte die Gruppen–Zugehörigkeit der Lodatei angepasst werden (z.B. auf die Gruppe `wheel`)

```
log_group = root
log_format = ENRICHED   # "enriched" or "raw"
```

- Beim Log–Format `enriched` werden die Log–Daten vom Audit–Daemon mit Metadaten versehen, welche die Auswertung der Log–Daten vereinfachen. Bei der Einstellung `raw` werden die Log–Daten wie vom Linux–Kernel gesendet gespeichert

Schreiben der Log–Datei

- Der Parameter `flush` zusammen mit dem Parameter `freq` bestimmt, wann die Log–Daten vom Audit–Daemon in die Logdatei geschrieben werden. In der Standard–Einstellung werden die Log–Daten spätestens nach 50 Meldungen im Puffer geschrieben.
    - Der Parameter `priority_boost` definiert die Prozess–Priorität des Audit–Daemon gegenüber normalen Benutzerprogrammen im System (vergleichbar mit einem negativen *nice* Wert) und verhindert, das der Audit–Daemon von einem Benutzerprogramm ausgebremst wird

```
flush = INCREMENTAL_ASYNC
freq = 50
priority_boost = 4
```

Anzahl Log-Dateien

- Der Audit-Daemon kann selbstständig die Log-Dateien *rotieren*, wenn die Log-Datei eine definierte Grösse erreicht.
  - Anders als bei anderen Log-Rotate Programmen (z.B. BIND 9) werden überschüssige Log-Dateien nur beim (Neu-)Start des Audit-Daemons, oder bei einem `space-left` Event gelöscht(!)

```
max_log_file = 8   # Max Groesse der Log-Datei in MB
num_logs = 5       # Anzahl Generationen der Log-Datei
max_log_file_action = ROTATE  # Aktion beim Erreichen der max. Log-Groesse
```

Log-Host Metadaten

- Der Audit-Log-Daemon kann die Log-Informationen um den Rechnernamen des Quell-Hosts anreichern
  - Dies ist bei remote Audit-Logging hilfreich
  - `name` ist der Hostname / Domain, welche den Log-Daten hinzugefügt wird wenn bei `name_format` der Wert `user` eingetragen wurde

```
name_format = NONE
##name = mydomain
```

Log-Host Metadaten

- Mögliche Werte für `name_format`:

Wert	Beschreibung
NONE	Keine Quell-Host Metadaten
HOSTNAME	Der Hostname des Systems (gethostname)
FQDN	Voller Domain-Name durch DNS Auflösung
NUMERIC	IP(v4) Adresse des Hosts
USER	Wert des `name` Parameters

E-Mail Meldungen

- Der Audit-Daemon kann bei außergewöhnlichen Systemzuständen (Fehler im Dateisystem oder in der Storage Hardware, kein Platz auf dem Speichermedium) E-Mail Warnungen an die Administratoren versenden
  - `action_mail_acct` ist ein lokaler E-Mail Benutzer, oder eine externe E-Mail Adresse. Ein MTA Programm muss unter `/usr/lib/sendmail` installiert sein
  - `verify_email` prüft ob der Domain-Name der angegebenen E-Mail Adresse per DNS aufgelöst werden kann

```
verify_email = yes
action_mail_acct = root
```

"Space Left" Event

- Der `space_left` Wert gibt an, ab welchem Limit an freiem Speicher auf dem Speichermedium der Log-Datei der Audit-Daemon eine Warnung absetzen soll
  - In der Standard-Einstellung wird die Warnung in das `syslog` geschrieben
  - Der Wert ist in Megabyte (MB) und für modernen System wahrscheinlich zu gering (Empfehlung: 1000 MB). Der Wert kann auch in Prozent angegeben werden (Beispiel: `25%`)

```
space_left = 75
space_left_action = SYSLOG
```

"Admin Space Left" Event

- Der `admin_space_left` Wert gibt an, ab welchem Limit an freiem Speicher auf dem Speichermedium der Log-Datei der Audit-Daemon seine Arbeitsweise ändern soll. Dieser Wert sollte geringer als `space_left` sein.
  - In der Standard-Einstellung stellt der Audit-Daemon seine Arbeit ein
  - Der Wert ist in Megabyte (MB) und für modernen System wahrscheinlich zu gering (Empfehlung: 300 MB). Der Wert kann auch in Prozent angegeben werden (Beispiel: `5%`)

```
admin_space_left = 50
admin_space_left_action = SUSPEND
```

Speicher-Medium Fehler Event

- Diese Werte geben an wie der Audit-Daemon auf Fehler beim Schreiben der Log-Datei reagieren soll
  - In der Standard-Einstellung stellt der Audit-Daemon seine Arbeit ein

```
disk_full_action = SUSPEND
disk_error_action = SUSPEND
```

Event-Aktionen des Audit-Daemon

Schlüsselwort	Beschreibung
ignore	Zustand ignorieren, keine Aktion
syslog	Zustand via `syslog` melden
rotate	Log-Dateien rotieren, überflüssige Log-Dateien entfernen
exec	Ein Skript ausführen

Schlüsselwort	Beschreibung
suspend	Keine Logdaten mehr schreiben
single	System in den Single–User–Mode versetzen, Netzwerk deaktivieren
halt	System herunterfahren (shutdown)

2.6.4. Weiterleiten von Event–Informationen

Audit–Daemon Plugins

- Der Audit–Daemon kann Audit–Events an andere Log–Systeme (z.B. SYSLOG oder SIEM Systeme) weiterleiten
- Hierzu werden Plugins geladen welche das jeweilige Logging–Protokoll implementieren
  - Die Plugins befinden sich unterhalb von `/etc/audit/plugins.d`
  - `max_restarts` gibt an wie oft ein Plugin nach einen Crash von Audit–Daemon neu gestartet wird

```
max_restarts = 10
plugin_dir = /etc/audit/plugins.d
```

Beispiel der Syslog–Plugin Konfiguration

- Um ein Plugin zu aktivieren wird der Wert `active` auf `yes` gesetzt und danach der Audit–Daemon neu gestartet

```
active = no
direction = out
path = /sbin/audisp-syslog
type = always
args = LOG_INFO
format = string
```

Weitere Konfigurations–Optionen

- Die weiteren Konfigurations–Optionen in der Datei `auditd.conf` werden für den Empfang von Audit–Log–Daten über das Netzwerk benötigt
- Wir werden uns diese Konfiguration später anschauen

2.6.5. Audit–Subsystem Status abfragen

Dienste Status

- Auf RedHat basierten Linux–Systemen ist der Audit–Daemon in der Regel vorinstalliert und gestartet (speziell wenn SELinux installiert und aktiviert ist)

```
# systemctl status auditd
● auditd.service - Security Auditing Service
     Loaded: loaded (/usr/lib/systemd/system/auditd.service; enabled; vendor preset:
enabled)
     Active: active (running) since Sun 2022-10-16 08:25:40 UTC; 12h ago
       Docs: man:auditd(8)
             https://github.com/linux-audit/audit-documentation
   Main PID: 645 (auditd)
      Tasks: 4 (limit: 2506)
     Memory: 8.1M
        CPU: 960ms
     CGroup: /system.slice/auditd.service
             ├─645 /sbin/auditd
             └─647 /usr/sbin/sedispatch

Oct 16 08:25:40 localhost augenrules[660]: enabled 1
Oct 16 08:25:40 localhost augenrules[660]: failure 1
Oct 16 08:25:40 localhost augenrules[660]: pid 645
Oct 16 08:25:40 localhost augenrules[660]: rate_limit 0
Oct 16 08:25:40 localhost augenrules[660]: backlog_limit 8192
Oct 16 08:25:40 localhost augenrules[660]: lost 0
Oct 16 08:25:40 localhost augenrules[660]: backlog 4
Oct 16 08:25:40 localhost augenrules[660]: backlog_wait_time 60000
Oct 16 08:25:40 localhost augenrules[660]: backlog_wait_time_actual 0
Oct 16 08:25:40 localhost systemd[1]: Started Security Auditing Service.
```

2.6.6. Ad-Hoc Auditing

Befehl autrace

- Mit dem Audit-Subsystem Trace Befehl `autrace` kann ein Audit-Trace eines Prozesses (hier 4 x ICMP Echo per `ping`) erzeugt werden
  - Der Trace ist unabhängig vom geladenen Audit-Regelwerk

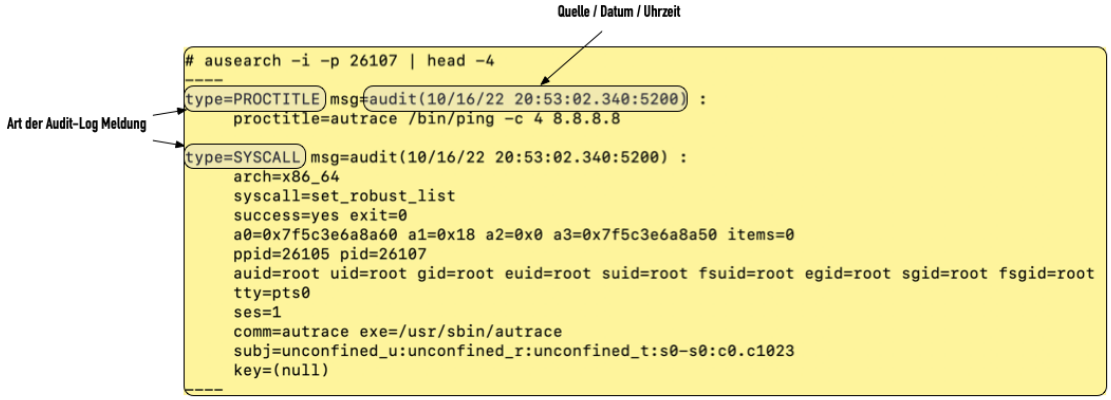
```
   # autrace /bin/ping -c 4 8.8.8.8
   Waiting to execute: /bin/ping
   PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
   64 bytes from 8.8.8.8: icmp_seq=1 ttl=60 time=1.11 ms
   64 bytes from 8.8.8.8: icmp_seq=2 ttl=60 time=0.807 ms
   64 bytes from 8.8.8.8: icmp_seq=3 ttl=60 time=0.666 ms
   64 bytes from 8.8.8.8: icmp_seq=4 ttl=60 time=0.726 ms
+
   --- 8.8.8.8 ping statistics ---
   4 packets transmitted, 4 received, 0% packet loss, time 3045ms
   rtt min/avg/max/mdev = 0.666/0.827/1.110/0.170 ms
   Cleaning up...
   Trace complete. You can locate the records with 'ausearch -i -p 26107'
```

Auditlog des Trace anzeigen

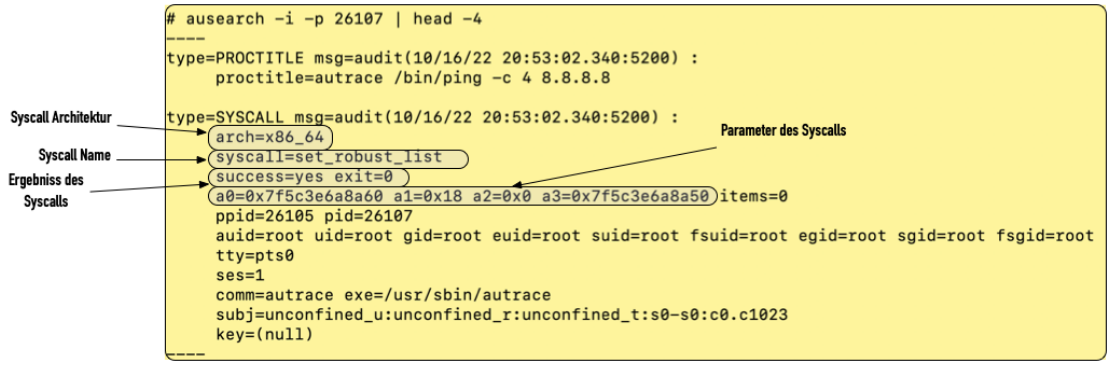
- Audit Log für einen speziellen `autrace` Aufruf anschauen

```
ausearch -i -p <audit-id>
```

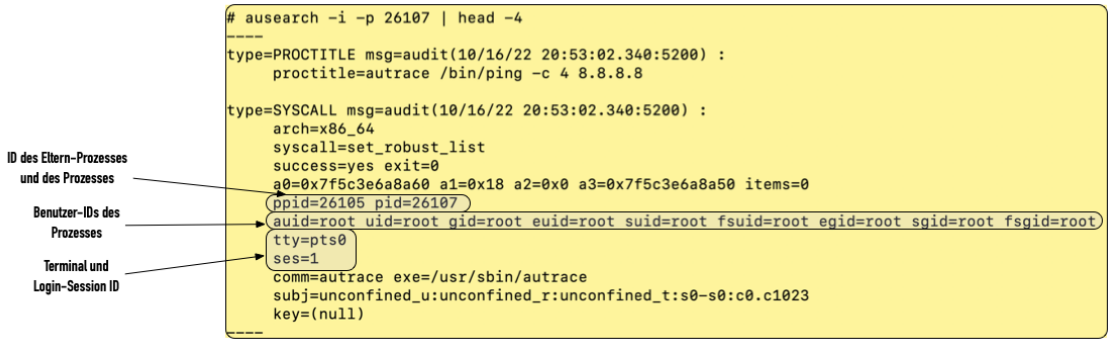
Anatomie eines Audit-Log Eintrags



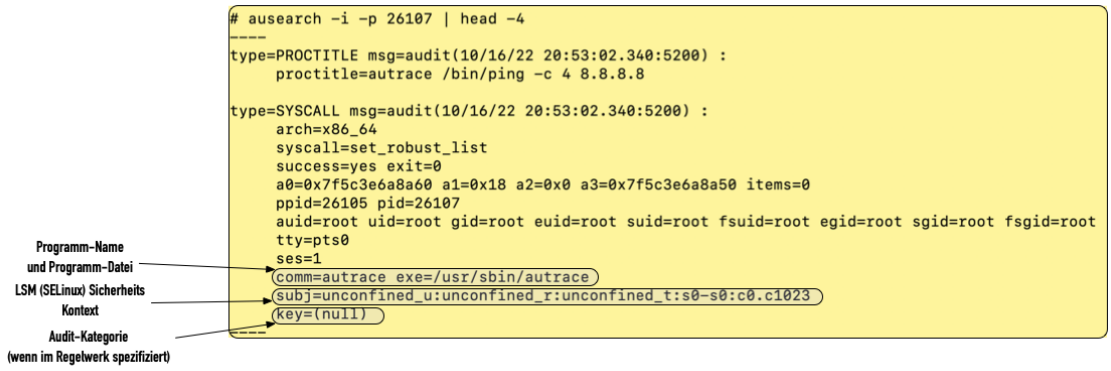
Anatomie eines Audit-Log Eintrags



Anatomie eines Audit-Log Eintrags



Anatomie eines Audit–Log Eintrags



2.6.7. Audit–Regelwerke

Audit Richtlinien Regelwerke

- Die Richtlinien–Regelwerke des Audit–Subsystems sind für den Einsatz von SELinux nicht notwendig, aber bieten oft eine sinnvolle Ergänzung zu den SELinux Funktionen
  - SELinux: Durchsetzen einer Richtlinie
  - Audit–Subsystem: Überwachung einer Richtlinie
- Die Richtlinien Regelwerke liegen im Verzeichnis `/etc/audit/rules.d/`
- Beim aktivieren der Regeln werden die Dateien in diesem Verzeichnis in der alphanumerischen Reihenfolge der Dateinamen zu einer Datei unter `/etc/audit/audit.rules` zusammengefasst und in den Linux–Kernel geladen
- Die Datei `/etc/audit/audit.rules` sollte nicht manuell editiert werden, Änderungen werden überschrieben

Beispiel Richtlinien Regelwerke

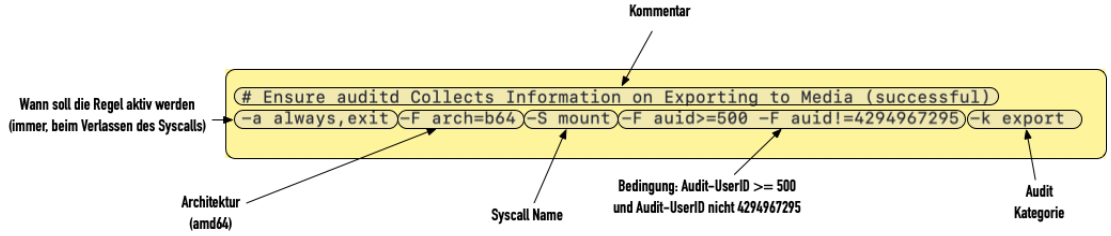
- Beispiel–Richtlinien werden unter `/usr/share/doc/auditd/examples/rules` mitgeliefert und können in `/etc/audit/rules.d/` kopiert werden
  - Die Regelwerke sollten jeweils manuell geprüft und ggf. angepasst werden (System–Architektur, Syscall–Namen)
  - Einige Regelwerke werden für das System individuell via Shell–Skripte erzeugt und müssen ggf. nach Software–Installationen oder –Updates neu generiert werden

Inhalt der Audit–Regelwerke

- Jede Zeile der Regelwerke ist eine Regel
- Die Regeln sind die Kommandozeilen–Parameter des Programms `auditctl`
- Die Dokumentation befindet sich in der `man` Page zu `auditctl`

```
# man auditctl
```

Beispiel einer Audit–Regel



Immutable vs. Mutable

- Audit–Regelwerke können nach dem Laden in den Kernel als *unveränderbar* (immutable) markiert werden

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

- Unveränderbare Regeln können nicht mehr im laufenden Betrieb des Linux–Systems geändert werden, es ist immer ein Neustart (Reboot) notwendig

2.6.8. Audit–Regelwerke aktivieren

augenrules

- Das Programm `augenrules` prüft die Regelwerke, fasst diese zusammen und lädt die Regeln in den Linux–Kernel

```
# augenrules --check
# augenrules --load
```

Status des Audit Subsystems

- 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
```

Aktuelles Regelwerk auflisten

- Aktive Regeln auflisten

```
# auditctl -l
-a always,exit -F arch=b32 -S stime,settimeofday,adjtimex -F key=time-change
-a always,exit -F arch=b64 -S adjtimex,settimeofday -F key=time-change
-a always,exit -F arch=b32 -S clock_settime -F a0=0x0 -F key=time-change
-a always,exit -F arch=b64 -S clock_settime -F a0=0x0 -F key=time-change
[...]
```

2.6.9. Audit-Logs auswerten

Abfragen mit "ausearch"

- Mittels des Programms `ausearch` lässt sich das lokale Audit-Log abfragen

Beispiel-Abfrage mit "ausearch"

- Alle Audit-Einträge zum Thema "sudo" zeigen

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

Beispiel-Abfrage mit "ausearch"

- Report über fehlgeschlagene Anmeldeversuche

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

Beispiel-Abfrage mit "ausearch"

- Alle Audit-Meldungen für Benutzer UID 1000

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

Beispiel-Abfrage mit "ausearch"

- Fehlgeschlagene Syscalls seit gestern

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

CSV Ausgabe

- `ausearch` erlaubt die Ausgabe der Abfrage-Ergebnisse als CSV-Datei
- Diese Dateien können in Office-Programme oder Datenbanken importiert werden

```
# ausearch --start today --format csv 2>/dev/null > audit-today.csv
```

Reports

- Der Befehl `aureport` bietet viele zusammengefasste Reports auf den Audit-Log-Dateien. Hier zum Beispiel ein Report über Verstöße gegen LSM (AVC) Richtlinien:

```
  # aureport -a
+
```

```
  AVC Report
  ============================================================
  # date time comm subj syscall class permission obj result event
  ============================================================
  1. 10/16/22 08:25:41 chronyd system_u:system_r:chronyd_t:s0 262 lnk_file read
system_u:object_r:unlabeled_t:s0 denied 22
  2. 10/17/22 00:01:01 logrotate system_u:system_r:logrotate_t:s0 262 file getattr
system_u:object_r:unlabeled_t:s0 denied 6127
  3. 10/17/22 00:01:01 logrotate system_u:system_r:logrotate_t:s0 262 file getattr
system_u:object_r:unlabeled_t:s0 denied 6128
```

Reports

- Weitere Reports

Befehl	Report
aureport –s	Syscall Report
aureport –p	Prozess Report
aureport –x	Report nach ausführbaren Dateien
aureport –f	Report nach Dateizugriffen
aureport –u	Report über Benutzeraktivitäten
aureport –l –i	Report über Logins

2.6.10. Audit–Hilfsprogramme

Spezielle Abfrageprogramme

- Die letzten Logins auf dem System anzeigen

```
# aulast
```

Letzte Anmeldezeiten von Benutzern

- Wann haben sich Benutzer zum letzten Mal am System angemeldet?

```
# aulastlog
```

Statistiken über das Audit–Log

- Zusammenfassung des Audit–Log

```
# aureport
```

Grafische Auswertungen

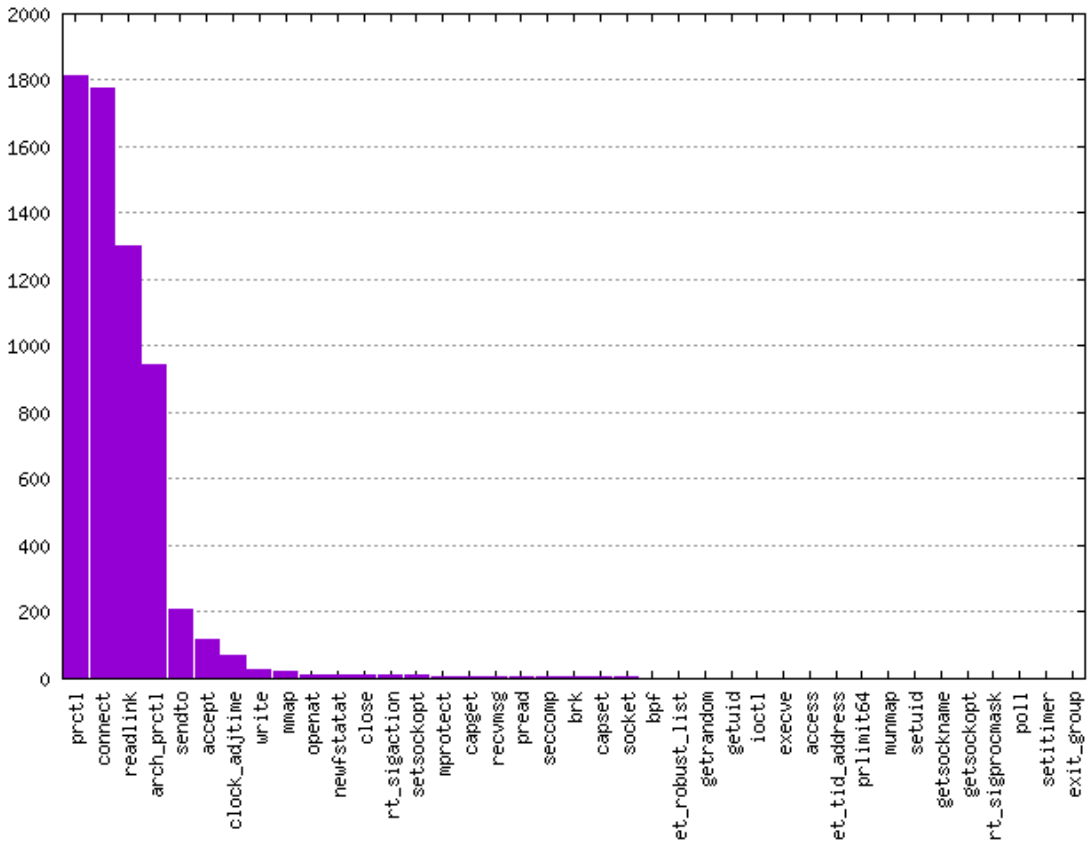
- Eine grafische Auswertung von Audit-Logs ist mit Skriptes des Linux-Audit Entwicklers Steve Grubb möglich
    - Diese Skripte wurden angepasst um mit modernen Versionen des Programms `gnuplot` zu funktionieren, und das Ausgabeformat wurde von Postscript auf PNG-Dateien geändert

```
$ sudo dnf install epel-release
$ sudo dnf 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
```

Beispiel einer Syscall Grafik



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 Dateien zu?

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

Weitere Audit-Logauswertungs-Programme

- AuditExplorer – An R shiny app that visualizes audit data using many tools all in one app: https://github.com/stevegrubb/audit-explorer/ [https://github.com/stevegrubb/audit-explorer/]

2.6.11. Vorlagen für Audit-Richtlinien

2.6.12. Vorlagen/Empfehlungen für Audit-Regeln

- CIS Benchmark Webseite: https://downloads.cisecurity.org/ [https://downloads.cisecurity.org/]
- A Linux Auditd rule set mapped to MITRE's Attack Framework https://github.com/bfuzzy1/auditd-attack [https://github.com/bfuzzy1/auditd-attack]
- Enhanced AuditD (Audit Daemon) rules for Linux systems, HIPAA & HITRUST compliance https://github.com/gnxsecurity/enhanced-auditd-rules [https://github.com/gnxsecurity/enhanced-auditd-rules]
- An auditd ruleset for monitoring linux servers https://github.com/benjaminkoffel/auditd-rules [https://github.com/benjaminkoffel/auditd-rules]
- UnderstandingAuditdRules – This is an overview of writing auditd rules for linux https://github.com/clevelandjosh/UnderstandingAuditdRules [https://github.com/clevelandjosh/UnderstandingAuditdRules]

2.6.13. Audit Subsystem Praxis

Konfiguration

- Passe die Audit-Daemon Konfiguration an. Benutze hierfür die Informationen aus der Beschreibung oben

```
$EDITOR /etc/audit/auditd.conf
```

- Audit-Dämon prüfen

```
systemctl status auditd
journalctl -u auditd
```

- Audit-Daemon starten und stoppen

```
service auditd stop
service auditd start
```

Ad-Hoc Trace von Programmen

- Starte einen Audit-Subsystem Trace eines Prozesses (hier 4 x ICMP Echo per `ping`)

```
autrace /bin/ping -c 4 8.8.8.8
```

- Audit Log für einen speziellen `autrace` Aufruf anschauen. Die genaue Kommandozeile wurde vom `autrace` Programm ausgegeben.

```
ausearch -i -p <audit-id>
```

- Die Log-Ausgaben können über eine Shell-Pipeline ausgewertet werden. In diesem Beispiel werden alle System-Calls des `autrace` Aufruf aufgelistet und nach Häufigkeit sortiert ausgeben:

```
ausearch -i -p <audit-id> | grep type=SYSCALL | cut -f 6 -d ' ' | sort | uniq -c | sort
-n
```

- Wurde der Hostname als Metadaten bei den Audit-Meldungen mit ausgegeben Konfiguration des Audit-Daemon), so muss per `cut` Befehl das Feld #7 ausgewertet werden (…`| cut -f 7 -d ' ' |`…)

Richtlinien-Dateien

- Die vorinstallierte (leere) Richtlinien-Datei löschen

```
rm /etc/auditd/rules.d/audit.rules
```

- Audit-Richtlinien Beispiele befinden sich unter `/usr/share/audit/sample-rules`
- Richtliniendateien aus den Beispielen in das Verzeichnis für die Audit-Regeln kopieren

```
cp /usr/share/audit/sample-rules/10-base-config.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/11-loginuid.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/30-stig.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/32-power-abuse.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/42-injection.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/43-module-load.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/70-einval.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/71-networking.rules /etc/audit/rules.d/
cp /usr/share/audit/sample-rules/99-finalize.rules /etc/audit/rules.d/
```

- Richtlinien für priviligierte Programme erstellen

```
cd /etc/audit/rules.d/
```

```
find /bin -type f -perm -04000 2>/dev/null | awk '{ printf "-a always,exit -F path=%s
-F perm=x -F auid>=1000 -F auid!=unset -F key=privileged\n", $1 }' > 31-
privileged.rules
find /sbin -type f -perm -04000 2>/dev/null | awk '{ printf "-a always,exit -F path=%s
-F perm=x -F auid>=1000 -F auid!=unset -F key=privileged\n", $1 }' >> 31-
privileged.rules
find /usr/bin -type f -perm -04000 2>/dev/null | awk '{ printf "-a always,exit -F
path=%s -F perm=x -F auid>=1000 -F auid!=unset -F key=privileged\n", $1 }' >> 31-
privileged.rules
find /usr/sbin -type f -perm -04000 2>/dev/null | awk '{ printf "-a always,exit -F
path=%s -F perm=x -F auid>=1000 -F auid!=unset -F key=privileged\n", $1 }' >> 31-
privileged.rules
filecap /bin 2>/dev/null | sed '1d' | awk '{ printf "-a always,exit -F path=%s -F
perm=x -F auid>=1000 -F auid!=unset -F key=privileged\n", $2 }' >> 31-privileged.rules
filecap /sbin 2>/dev/null | sed '1d' | awk '{ printf "-a always,exit -F path=%s -F
perm=x -F auid>=1000 -F auid!=unset -F key=privileged\n", $2 }' >> 31-privileged.rules
filecap /usr/bin 2>/dev/null | sed '1d' | awk '{ printf "-a always,exit -F path=%s -F
perm=x -F auid>=1000 -F auid!=unset -F key=privileged\n", $2 }' >> 31-privileged.rules
filecap /usr/sbin 2>/dev/null | sed '1d' | awk '{ printf "-a always,exit -F path=%s -F
perm=x -F auid>=1000 -F auid!=unset -F key=privileged\n", $2 }' >> 31-privileged.rules
```

- Die Richtlinie kompilieren und prüfen

```
augenrules --check
```

- Die kombinierte Audit-Richtlinien-Datei anschauen

```
less /etc/audit/audit.rules
```

- Die neue Policydatei in den Kernel laden

```
augenrules --load
```

- Aktive Audit-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
```

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 Audit-Regeln für Systemd:

- 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`)

2.7. LSM – LINUX-SICHERHEITS-MODULE

- LSM (Linux Security Module) sind Erweiterungen des Linux-Kernels
  - Der Linux-Kernel definiert Schnittstellen, in welche sich die LSMs einklinken können:
    - Syscalls
    - Dateizugriffe
    - Prozess-Erstellung
    - Namespaces und cgroups
    - Benutzer-Identität (UID/GID)
    - ...
  - Ein LSM Modul kann sich in eine oder mehrere dieser Schnittstellen einklinken
  - Wird diese Kernel-Funktion benutzt, so wird das LSM aktiv und wird die Kernel-Funktion nach Prüfung erlauben oder verbieten
- Link: A Brief Tour of Linux Security Modules [https://www.starlab.io/blog/a-brief-tour-of-linux-security-modules/]
- Major LSMs: Mandatory Access Controls – nur eines dieser LSM kann (derzeit) im Linux-Kernel

aktiviert sein

- SELinux
- AppArmor
- SMACK (Simplified Mandatory Access Control)
- TOMOYO

- Minor LSMs: diese LSMs können zusätzlich zu den Major-LSMs und anderen Minor-LSMs aktiviert sein
  - YAMA
  - LoadPin
  - SafeSetID
  - Lockdown
  - Landlock
  - BPF – LSM Sicherheitsrichtlinien können mittels eBPF durchgesetzt werden
  - capabilities – Linux capabilities

2.7.1. Prüfen, welche LSMs im aktuell geladenen Linux-Kernel aktiv sind

```
# cat /sys/kernel/security/lsm
lockdown,capability,yama,tomoyo,bpf
```

2.7.2. YAMA

- YAMA sammelt Sicherheitsfunktionen aus Linux-Kernel-Sicherheitspatches, welche nicht in eine der anderen Linux-Sicherheits-Module passen
- in aktuellen Linux-Kerneln bietet mit YAMA einen Schutz gegen **ptrace** (ptrace_scope), d.h. das ein Prozess den Speicher und die Ausführung eines anderen Prozesses überwachen kann (diese Funktion wird für die Benutzung von Debuggern bei der Programmentwicklung benötigt). Auf Produktions-Serversystemen ist diese Funktion oft nicht benötigt.
- Werte für `ptrace_scope`

Wert	Beschreibung +
0	normales Verhalten, jeder Prozess und Benutzer kann andere Prozesse überwachen
1	ein Prozess kann nur einen eigenen Kind-Prozess überwachen
2	nur ein Systemadministrator mit CAP_SYS_PTRACE kann Prozesse überwachen

3	keine Überwachen von Prozessen mittels *ptrace* möglich, diese Einstellung kann im laufenden System nicht überschrieben werden

Beispiel (Firefox als nutzerXX gestartet):

```
root$ dnf install strace
nutzerXX$ strace -p $(pgrep firefox)
root$ echo "1" > /proc/sys/kernel/yama/ptrace_scope
nutzerXX$ strace -p $(pgrep firefox)
strace: attach: ptrace(PTRACE_ATTACH, 3135): Operation not permitted
```

2.7.3. LoadPin

- LoadPin ist seit Kernel 4.7 Bestandteil von Linux
- LoadPin gewährleistet das alle Kernel-Daten (Module, Firmware-Blobs etc) vom gleichen, read-only Datei-System gelesen werden.
- LoadPin verhindert Angriffe auf den Kernel durch bösartige Kernel-Module (z.B. Root-Kits)
- Dokumentation:
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/LoadPin.rst [https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/LoadPin.rst]

2.7.4. SafeSetID

- Das *SafeSetID* Modul überwacht die Benutzung der SetUID/SetGID Syscalls (UID oder GID eines Prozesses ändern, z.B. via `su` oder `sudo` oder per SetUID-Bit in den Dateisystemrechten) und prüft die Benutztung gegen eine systemweite Whitelist
  - Mittels *SafeSetID* können Prozesse die Capability CAP_SETUID/CAP_SETGID benutzen um von einem unpriviligierten Benutzeraccount in einen anderen zu wechseln (um Rechte *abzugeben*), ohne das diese Programme die Rechte ausweiten können oder sogar *Root*-Rechte erlangen können
  - Die Whitelist wird als Text-Datei mit je einer Regel pro Zeile in das `securityfs` Pseudo-Dateisystem unter dem Pfad `safesetid/uid_allowlist_policy` (für UID Übergänge) und `safesetid/gid_allowlist_policy` (für GID Übergänge) geschrieben
  - Das Format jedes Eintrages ist `<ausgangs-UID>:<neue-UID>\n`
- Dokumentation:
  https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/SafeSetID.rst [https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/SafeSetID.rst]

2.7.5. Lockdown

- Das Lockdown Modul beschränkt bestimmte Funktionen des Linux Kernels (Laden von unsignierten Modulen, PTRACE, eBPF, Hibernation = Speicherinhalte auf Datenträger schreiben, Zugriffe auf spezielle Gerätedateien unter `/dev`, Kernel `perf` Schnittstellen, ACPI-Tabellen ...)

    - Lockdown kann per Kernel–Commandozeile aktiviert werden: Parameter `lockdown=`
    - Nachträglich kann Lockdown über das Pseudo–Dateisystem unter `/sys/kernel/security/lockdown` ein– oder aus–geschaltet werden
    - Lockdown kennt zwei Modi
        - `integrity` – Funktionen sind ausgeschaltet, welche den laufenden Kernel verändern können
        - `confidentiality` – Funktionen aus dem Modus `integrity` sind ausgeschaltet, plus Funktionen über welche sensible Daten aus dem Kernel ausgelesen werden können (z.B. Zugriffe auf `/dev/mem`)
- Lockdown LSM ist seit Kernel 5.4 verfügbar
- Lockdown Status ausgeben (hier `none`, Modi `integrity` und `confidentiality` sind möglich)

```
# cat /sys/kernel/security/lockdown
[none] integrity confidentiality
```

- Links
    - Lockdown as a security module [https://lwn.net/Articles/791863/]
    - Linux kernel lockdown, integrity, and confidentiality [https://mjg59.dreamwidth.org/55105.html]

2.7.6. Landlock

- Landlock ist ein LSM welches Prozessen erlaubt, die eigenen Dateisystem–Rechte (und die Rechte von Kind–Prozessen) über die Unix–Dateisystemrechte hinaus einzuschränken.
    - Da die Landlock Rechte–Einschränkungen vererbt werden, können auch *Supervisor* Programme erstellt werden, um Kind–Prozesse zu beschränken, ohne diese Kind–Prozesse im Quellcode ändern zu müssen
    - Landlock ist von der Idee vergleichbar mit dem OpenBSD `pledge` Mechanismus, wirkt jedoch nur auf Dateisystemrechte (`pledge` wirkt auf verschiedenen Syscall–Gruppen)
- Seit Kernel 5.13 verfügbar
- Dokumentation: https://landlock.io/ [https://landlock.io/]
- Kernel Dokumentation: https://www.kernel.org/doc/html/latest/security/landlock.html [https://www.kernel.org/doc/html/latest/security/landlock.html]

2.7.7. TOMOYO

- TOMOYO LSM kann zur Analyse eines Linux–Systems, oder zur Implementation einer Sicherheitrichtlinie für das Linux–System verwendet werden
- TOMOYO beschränkt Linux–Prozesse auf Basis des Dateisystem–Pfads und der Prozess–Hierarchy ("Domain" in der TOMOYO Terminology, hat aber keinen Zusammenhang mit DNS, dem Domain–Name–System)
- TOMOYO fasst einen oder mehrere Prozesse zu einer Domain zusammen
    - Jede Domain wird über ein Profil reglementiert

    - Es können 255 unterschiedliche Profile im TOMOYO LSM definiert werden
    - Jedes Profil kann unabhängig von anderen Profilen in einem von 3 Modi verwendet werden
        - Learning: Syscall–Aufrufe und Dateisystem–Interaktionen werden protokolliert und in das Profil aufgenommen (Erstellung eines Baseline–Profils für eine Prozess–Gruppe/Domain)
        - Permissive: Verstösse gegen die Policy werden protokolliert, aber nicht durchgesetzt
        - Enforcing: Verstösse gegen die Policy werden aktiv verhindert und protokolliert
- Kernel–Dokumentation: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin–guide/LSM/tomoyo.rst [https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin–guide/LSM/tomoyo.rst]
- Handbuch: https://tomoyo.osdn.jp/2.6/index.html.en [https://tomoyo.osdn.jp/2.6/index.html.en]
- TOMOYO kann alternativ auch als Linux–Kernel–Modul geladen werden (dann ist es kein LSM, sondern kann zusätzlich zu anderen Major–LSMs benutzt werden)

2.7.8. SMACK

- SMACK ist ein Mandatory Access Controll System im Linux Kernel, welches im Vergleich zu SELinux weniger komplex ist
- SMACK ist seit Kernel 2.6.25 Teil des Linux–Kernels
- SMACK wird heute hauptsächlich in Linux–Systemen von IoT–Geräten und in Automotive–Anwendungen verwendet (z.B. im Tizen OS von Samsung)
    - Wie auch SELinux arbeitet SMACK auch mit Sicherheits–Label auf Basis von erweiterten Attributen in den Linux–Dateisystemen
        - Es ist möglich SMACK in mittels der eingebauten Sicherheitspolicy auch ohne erweiterte Attribute zu betreiben, dann kann die Sicherheitsrichtlinie jedoch nicht angepasst werden
    - SMACK bindet Sicherheitslabel beim Empfang an IP–Pakete
        - Die Sicherheitsrichtlinie kann benutzt werden, um die Kommunikation von Anwendungen auf bestimmte IP–Adressen oder Netzwerke zu beschränken
        - Diese Funktion von SMACK kann zu Performance–Verlusten bei der Netzwerkkommunikation führen, wenn SMACK im Kernel aktiv ist
- SMACK Kernek Dokumentation [https://www.kernel.org/doc/html/v4.14/admin–guide/LSM/Smack.html]
- Heise iX: SMACK im Linux Kernel [https://www.heise.de/select/ix/2017/3/1488028406202537]

2.7.9. SELinux

- SELinux (Security–Enhanced Linux; engl. „sicherheitsverbessertes Linux“) ist eine Erweiterung des Linux–Kernels. Es implementiert die Zugriffskontrollen auf Ressourcen im Sinne von Mandatory Access Control. SELinux wurde von der NSA für eigene Bedürfnisse entwickelt und wird von dem Linux–Distributor Red Hat gepflegt.
- SELinux ist Open–Source–Software und setzt sich aus einem Kernel–Modul, Hilfsprogramme und aus zahlreichen Erweiterungen für Systemprogramme zusammen.

- Kernbestandteil von SELinux sind die Policies, welche sehr detailliert beschreiben, welche Zugriffe (Dateisystem, Syscalls, Netzwerk) einem Prozess oder einem Benutzer erlaubt sind.
- SELinux kann mit verschiedenen Policy–Einstellungen betrieben werden:
  - `full` – alle Ressourcen und Anwendungen im Linux–System sind von der SELinux Policy abgedeckt. Dies ist sehr aufwändig und wird von den Linux–Distributionen **nicht** angeboten
  - `targeted` – SELinux Policies existieren für kritische Komponenten im Linux–System (z.B. Systemd, Webserver, Mailserver etc). Nur diese Komponenten sind von SELinux abgesichert, alle anderen Komponenten befinden sich im `unconfined` Modus und werden von SELinux **nicht** beschränkt. Die Linux–Distributionen bieten Policies für den `targeted` Modus an.

SELinux erkunden

- wir arbeiten auf einer Rocky Linux VM im Internet. Benutzername `user` und Passwort `villa`, Benutzer `root` kann durch `sudo` erreicht werden
- SELinux Hilfspakete installieren

```
  yum install policycoreutils setools libselinux-utils selinux-policy-doc setools-
console
    \ policycoreutils-python3 selinux-policy-devel policycoreutils-newrole
```

- SELinux Label (Context) auf Dateien/Prozesse/Benutzer anzeigen

```
  ls -lZ <pfad>
  ps -auxZ
  id -Z
```

- SELinux Status abfragen

```
  getenforce
  sestatus
  sestatus -v
```

- SELinux Module auflisten

```
  semodule -l | less
```

- Modul–Dateien (binär) und die fertige Policy

```
  ls -l /etc/selinux/targeted/active/modules/100/
  ls -lh /etc/selinux/targeted/policy/
```

- SELinux Policy Quelldateien

```
  ls -l /usr/share/selinux/devel/
```

- Apache Webserver installieren und starten

```
  yum install httpd
  systemctl enable --now httpd
```

- Kleine Webseite anlegen

```
$EDITOR /var/www/html/index.html
```

- Inhalt der HTML-Datei

```
<html>
<body>
<h1>Apache Webserver</h1>
</body>
</html>
```

- SELinux Security Context auf der Datei

```
ls -lZ /var/www/html/index.html
```

Ein Problem für SELinux erzeugen und bereinigen

- `index-html` Datei im Verzeichnis des Benutzers `root` erstellen und dann in das Apache-WWW-Verzeichnis verschieben:

```
rm /var/www/html/index.html
$EDITOR /root/index.html
mv /root/index.html /var/www/html/index.html
ls -lZ /var/www/html/index.html
```

- Apache sollte nun nicht mehr in der Lage sein, die HTML-Datei auszuliefern (Default-Apache 2 Webseite erscheint)
- SELinux Meldungen im Audit-Log

```
ausearch -m avc -ts recent -c httpd -i
```

- SELinux Security Context prüfen

```
matchpathcon -V /var/www/html/index.html
```

- SELinux Security Context anpassen

```
chcon --type httpd_sys_content_t /var/www/html/index.html
```

- (Alternativ) SELinux Security Context aus der SELinux Policy angleichen

```
restorecon -v /var/www/html/index.html
```

- SELinux Security Context für Apache

```
sesearch --allow --source httpd_t --target httpd_sys_content_t --class file
```

- weitere SELinux Tools

```
seinfo       # Übersicht über das SELinux Regelwerk
seinfo -u    # Übersicht der SELinux Benutzer
seinfo -r    # Übersicht der SELinux Rollen
seinfo -t    # Übersicht der SELinux (Datei-) Typen
```

SELinux und Benutzer

- Benutzer `user` in die Benutzerklasse `user_u` einfügen

```
  semanage login -l
  semanage user -l
  semanage login -a -s user_u user
  # su oder sudo sollten nun für den Benutzer nicht mehr möglich sein
  cat /etc/selinux/targeted/seusers
  semanage login -a -s guest_u user
  getsebool allow_guest_exec_content
  setsebool allow_guest_exec_content off
  # scripts sind nun nicht mehr direkt ausführbar (indirekt über BASH oder SH
funktioniert es trotzdem)
```

- Fehlersuche bei SELinux Problemen

```
  ausearch -m avc -ts recent
```

Aufgabe: Apache auf einem anderen Port als 80 oder 443

- Schritte auf der VM ausführen
- SELinux auf `enforcing` schalten: `setenforce 1`
- Apache Installieren `dnf install httpd`
- Ändere die Apache Konfiguration unter `/etc/httpd/conf/httpd.conf` so das der Apache auf Port 1235/TCP auf HTTP-Anfragen horcht (Konfigurationsparameter `Listen: 1235`)
- Port 1235 in der Firewall freischalten

```
  # firewall-cmd --zone=public --add-port=1235/tcp --permanent
  # firewall-cmd --reload
```

- Versuche den Apache Webserver mittels systemd neu zu starten (`systemctl start httpd`). Dies wird fehlschlagen, da `httpd` sich nicht auf Port 1235 binden kann.
- Schaue in per `ausearch` in die Audit-Logdatei nach SELinux Fehlern des Prozesses `httpd` (`ausearch -m avc -i -ts recent -c httpd`)
- Benutze den Befehl `semanage port` um den Port `1235` für den Prozess `httpd` im SELinux freizuschalten und starte den Apache Webserver neu.
- Teste, on der Firefox-Browser unter der URL http://selinuxNNN.linux-sicherheit.org:1235/ [http://selinuxNNN.linux-sicherheit.org:1235/] die Webseite des Webservers sehen kann.

Beispiellösung

- Apache Konfiguration bearbeiten

```
$EDITOR /etc/httpd/conf/httpd.conf
```

- Apache Prozess neu starten. Dies sollte fehlschlagen, da SELinux die Benutzung von Port 1235 für den Prozess `httpd` verbietet

- SELinux Fehler für `httpd` im Audit-Log suchen

```
ausearch -m avc -c httpd -ts recent -i
```

- Port 1235 für Apache in der SELinux Richtlinie freischalten

```
semanage port -a -t http_port_t -p tcp 1235
```

- Apache Webserver neu starten

```
systemctl restart httpd
```

- Änderungen mit `semanage` sind persistent, die Port-Änderung ist unter `/var/lib/selinux/targeted/active/ports.local` abgespeichert:

```
# This file is auto-generated by libsemanage
# Do not edit directly.

portcon tcp 1235 system_u:object_r:http_port_t:s0
```

CHAPTER 3. TAG 3

3.1. SELINUX (TEIL 2)

3.1.1. SELinux Aufgabe: Apache Dokument-Root

- Erstelle ein Verzeichnis für HTML Seiten unter `/srv/websites`
- Erstelle in dem Verzeichnis eine **einfache** `index.html` Datei
- Ändere die Konfigration des Apache Webservers (siehe vorherige Aufgabe) so das der Document-Root auf das Verzeichnis `/srv/websites` zeigt
- Starte den Apache Webserver neu und teste ob die Webseite ausgeliefert wird (sollte nicht)
- Prüfe die SELinux Meldungen im Audit-Log mit `ausearch -m avc -i -ts recent`
- Benuzte den Befehl `semanage fcontext` (MAN-Page lesen) um die SELinux-Label `httpd_sys_content_t` für das Verzeichnis `/srv/websites` zu definieren
- Benutze den Befehl `restorecon` um den Dateien unter `/srv/website` mit neuen SELinux Label zu versehen
- Erstelle eine neue HTML-Datei unter `/srv/websites`, prüfe das SELinux Context-Label dieser neuen Datei mit `ls -lZ`
- Teste das die Webseiten nun korrekt ausgeliefert werden

Lösung:

- Apache Konfiguration

```
DocumentRoot "/srv/websites"
<Directory "/srv/websites">
    AllowOverride None
    # Allow open access:
    Require all granted
</Directory>

# Further relax access to the default document root:
<Directory "/srv/websites">
[...]
```

- SELinux File-Context anpassen

```
semanage fcontext -a -t httpd_sys_content_t "/srv/websites(/.*)?"
```

- Datei-Label anpassen

```
restorecon -vr /srv/websites
```

3.1.2. Durchsetzung der Richtlinie für Module ausschalten

- Es ist möglich einzelne SELinux Module in einen *permissive* Modus zu versetzten
    - In diesem Modus wird die SELinux Policy für dieses Modul nicht mehr vom Kernel

durchgesetzt
    - Verstösse gegen die Policy werden jedoch weiterhin im Audit–Log protokolliert
- Ein Modul (hier `httpd_t` für Apache oder NGINX Webserver) in den *permissive* Modus setzen

```
semanage permissive -a httpd_t
```

- Alle Module auflisten, welche im *permissive* Modus laufen

```
# semanage permissive -l
```

- Permissive Modus von einem Modul entfernen

```
semanage permissive -d httpd_t
```

- Permissive Modus von **allen** Modulen entfernen

```
semanage permissive -D
```

3.1.3. SELinux – Policy Development

- Andere Web–Server (Apache/NGINX etc) stoppen
- Für die Dauer dieser Übung die Firewall deaktivieren

```
systemctl stop firewalld
```

SELinux Policy erstellen

- In diesem Kapitel werden wir eine SELinux Richtlinie (Policy) für einen Dienst entwickeln, welcher bisher noch nicht durch SELinux geschützt ist
    - Dies kann in der Praxis für Software notwendig werden, welche von externen Entwicklern geliefert wurde oder im eigenen Hause entwickelt wird
    - Unsere Beispiel–Anwendung ist ein sehr einfacher Webserver
- Um die Beispiel–Anwendung aus dem Quellcode übersetzen zu können installieren wir den GCC C–Compiler

```
dnf install gcc
```

- Für die Entwicklung neuer SELinux Richtlinien benötigen wir die Pakete für die SELinux Policy–Entwicklung (diese sind im Kurs ggf. schon installiert)

```
dnf install policycoreutils-python3 selinux-policy-devel
```

- Nach der Installation dieser Pakete befinden sie die SELinux Policy Quelldateien (der von Red Hat und der Community erstellen Richtlinien) im Verzeichnis `/usr/share/selinux/devel/`

```
  ls -l /usr/share/selinux/devel/
  ls -l /usr/share/selinux/devel/include/contrib/
```

Unser Dienst: Ein einfacher Web-Server

- Hier ist der Quellcode (C Programmiersprache) eines (sehr) einfachen Web-Servers. Dieser Webserver liefert nur eine statische Webseite aus (diese Webseite ist fest im Quellcode des Servers eingebaut und wird nicht aus dem Dateisystem geladen):

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <err.h>

char response[] = "HTTP/1.1 200 OK\r\n"
"Content-Type: text/html; charset=UTF-8\r\n\r\n"
"<!DOCTYPE html><html><head><title>Bye-bye baby bye-bye</title>"
"<style>body { background-color: #111 }"
"h1 { font-size:4cm; text-align: center; color: black;"
" text-shadow: 0 0 2mm red}</style></head>"
  "<body><h1>Goodbye, world!</h1></body></html>\r\n";

int main()
{
  int one = 1, client_fd;
  struct sockaddr_in svr_addr, cli_addr;
  socklen_t sin_len = sizeof(cli_addr);

  int sock = socket(AF_INET, SOCK_STREAM, 0);
  if (sock < 0)
    err(1, "can't open socket");

  setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &one, sizeof(int));
  int port = 8080;
  svr_addr.sin_family = AF_INET;
  svr_addr.sin_addr.s_addr = INADDR_ANY;
  svr_addr.sin_port = htons(port);

  if (bind(sock, (struct sockaddr *) &svr_addr, sizeof(svr_addr)) == -1) {
    close(sock);
    err(1, "Can't bind");
  }

  listen(sock, 5);
  while (1) {
    client_fd = accept(sock, (struct sockaddr *) &cli_addr, &sin_len);
    printf("got connection\n");

    if (client_fd == -1) {
      perror("Can't accept");
      continue;
    }
```

```
    write(client_fd, response, sizeof(response) - 1); /*-1:'\0'*/
    close(client_fd);
  }
}
```

- Wir erstellen ein neues Verzeichnis für unser Project und erstellen die Datei mit dem Quellcode mit Hilfe eines Text-Editors (emacs, mg, nano, vim etc)

```
mkdir ~/src
cd ~/src
$EDITOR simple-server1.c
```

- Im nächsten Schritt wird der Quellcode in eine Programm-Datei übersetzt (`simple-server`)

```
gcc -o simple-server simple-server1.c
```

- Die Programmdatei kopieren wir in das Verzeichnis `/usr/local/bin`

```
cp simple-server /usr/local/bin
```

- Wir starten den Server im Hintergrund und testen die Funktion in dem wir uns mit einem Web-Browser an Port `8080` verbinden. Wir sollten dort eine "Hello^H^H^H^HGoodbye World" Meldung sehen. Bei jeder Verbindung gibt der Server den Text *Got connection* aus.

```
/usr/local/bin/simple-server &
```

- Wenn wir uns die SELinux Label des Dienstes anzeigen lassen sehen wir das dieses Dienst `unconfined` ist, also nicht durch SELinux abgesichert

```
ps -eZ | grep simple
ls -lZ /usr/local/bin/simple-server
```

Initiales SELinux Policy Modul

- Wir erstellen ein neues Verzeichnis fuer das neue SELinux Policy Modul

```
mkdir ~/selinux-src
cd ~/selinux-src
```

- Der Befehl `sepolicy generate` erzeugt eine Vorlage für ein SELinux Policy Modul

```
sepolicy generate -n simple-server --init /usr/local/bin/simple-server
```

- Es werden drei SELinux Policy Quelldateien erstell
  - `simple-server.te` – Type Enforcement Quellcode – auf welche Datei-Typen darf der Prozess zugreifen
  - `simple-server.fc` – File Context Quellcode – welche Datei-Typen werden benutzt (und in welchen Pfaden liegen diese)
  - `simple-server.if` – Interface Quellcode – Definiert die Übergänge zwischen den Dateisystem und Prozess Typen, und definiert die Regeln für die Richtlinien

    - `simple-server_selinux.spec` – Quelldatei für ein RPM Paket
    - `simple-server.sh` – Shell Skript zum bauen des RPM Pakets des SELinux Policy Moduls
- Diese Dateien können wir uns anschauen
- Die Policy ist im `permissive` Modus!

```
less simple-server.te
less simple-server.fc
less simple-server.if
```

- Wir testen diese SELinux Policy indem wir diese übersetzen und ein RPM-Paket erstellen. Das Paket `rpm-build` wird benötigt um ein RPM-Paket zu bauen

```
dnf -y install rpm-build
sh simple-server.sh
ls -l
```

- Die neue SELinux Policy befindet sich in der Datei `simple-server.pp`. Dieses neue SELinux Policy-Modul kann nun geladen und aktiviert werden

```
semodule -i simple-server.pp
```

- Die neue Policy wirkt sich nicht auf schon gestartete Prozesse aus. Daher stoppen wir den vorher gestarteten `simple-server` Prozess

```
pkill simple-server
```

- Wir erstellen eine SystemD Service-Unit für den Server Dienst

```
$EDITOR /etc/systemd/system/simple-server.service
```

- Die Unit-Datei

```
[Unit]
Description=a simple http server
After=syslog.target network.target

[Service]
ExecStart=/usr/local/bin/simple-server

[Install]
WantedBy=multi-user.target
```

- Die neue Systemd-Service-Datei und das `simple-server` Programm muss mit dem richtigen SELinux Label versehen werden

```
restorecon -R -v /etc/systemd/system/simple-server.service
restorecon -R -v /usr/local/bin/simple-server
```

- Systemd Service-Units neu laden und den Simple-Server starten

```
systemctl daemon-reload
systemctl start simple-server
systemctl enable simple-server
```

```
systemctl status simple-server
```

- Nun den Dienst benutzen (Mit dem Web-Browser auf Port 8080 zugreifen). Das SELinux Modul ist noch im Permissive Mode, Verstösse gegen die Policy werden im Audit-Log protokolliert

```
ausearch -m avc -ts recent -c simple-server
```

- Auch das Systemd-Journal liefert Fehlermeldungen über SELinux-Troubleshoot Modul `setroubleshoot`

```
journalctl | grep setroubleshoot
```

- Erklärungen zu den Policy-Fehlermeldungen ausgeben

```
ausearch -m avc -ts today -c simple-server | audit2why  | less
```

- Mittels des Programms `audit2allow` lassen sich Policy-Regeln aus den Audit-Meldungen erstellen. Diese Regeln sind selten 100% korrekt und müssen oft nachbearbeitet werden, helfen aber enorm bei der Erstellung eines Regelwerkes
    - Der Befehl `sepolgen-ifgen` erzeugt aus den SELinux Interface-Dateien des Systems Hilfdateien für die Erstellung der Policy-Dateien
    - Der Befehl `audit2allow -R` gibt die SELinux Policy-Regeln auf dem Terminal aus. Diese bauen wir per copy-n-paste in die Type-Enforcement-Quelldatei `simple-server.te` ein. Dabei muss auf die korrekte Reihenfolge der Abschnitte geachtet werden (`require` Block unter Deklarationen, `allow` Ausdrücke darunter):

```
sepolgen-ifgen -v
ausearch -m avc -ts today -c simple-server | audit2allow -R
require {
        type simple-server_t;
        class tcp_socket { bind create setopt accept listen };
}

#============= simple-server_t ==============
allow simple-server_t self:tcp_socket { bind create setopt accept listen };
corenet_tcp_bind_generic_node(simple-server_t)
corenet_tcp_bind_http_cache_port(simple-server_t)
```

- Neue Policy-Regeln in die Policy einfügen, Modul entfernen, neu kompilieren und dann neu laden

```
semodule -r simple-server
sh ./simple-server.sh
semodule -i simple-server.pp
systemctl restart simple-server
```

- Die Anwendung benutzen und testen, danach wieder das Audit-Log auf SELinux Fehler des `simple-server` Prozesses prüfen. GGf. neue Regeln erstellen und Modul und Programm neu laden
- Diese Schritte wiederholen bis keine SELinux Meldungen mehr im Audit-Log auftauchen

```
  ausearch -m avc -ts recent -c httpd -i
  <no matches>
```

- Die Anwendung ggf. für eine gewisse Zeit in Produktion im *permissive* Modus betreiben und auf SELinux Fehler prüfen
- Treten keine Fehler mehr auf, dann die Zeile `permissive simple-server_t;` in der Type-Enforcement Datei auskommentieren und das Modul im *enforcing* Modus betreiben
- Schalten wir nun das `simple-server` Modul in den *enforcing* Modus, werden wir feststellen das das Programm doch nicht wie gewünscht funktioniert
- Ein Trace des laufenden Programms mittels `strace` oder *eBPF* oder *bpftrace* zeigt das die Syscalls `shutdown` und `write` fehlschlagen. Diese werden von SELinux unterbunden, aber nicht an das Audit-Subsystem gemeldet.
- Aufgabe: Trage die Syscalls `shutdown` und `write` in die Policy ein, übersetze die Policy und teste erneut

3.1.4. Die "DoNotAudit" Regeln ausschalten

- Entwickler von SELinux-Policies können bestimmte Regeln vom Auditing ausschliessen
  - Um übermässiges Logging im Audit-Log zu vermeiden
  - Verstösse gegen SELinux-Regeln, welche mit `donotaudit` markiert sind, werden **nicht** im Audit-Log vermerkt
  - Bei der Entwicklung von neuen SELinux Policies kann dies stören, denn hier möchte man im Audit-Log ein möglichst vollständiges Bild aller Verstösse bekommen
  - Die `donotaudit` Regeln in der SELinux-Richtline ausschalten

```
# semodule -DB
```

- Um die `donotaudit` Regel wieder zu aktivieren

```
# semodule -B
```

3.2. SELINUX RESSOURCEN

Linux LSM

- Loadpin Dokumentation: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/LoadPin.rst [https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/LoadPin.rst]
- SafeSetID Dokumentation https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/SafeSetID.rst [https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/SafeSetID.rst]
- Lockdown Links
  - Lockdown as a security module [https://lwn.net/Articles/791863/]
  - Linux kernel lockdown, integrity, and confidentiality [https://mjg59.dreamwidth.org/55105.html]
- Landlock Links

    - Webseite: https://landlock.io/ [https://landlock.io/]
    - Kernel Dokumentation: https://www.kernel.org/doc/html/latest/security/landlock.html [https://www.kernel.org/doc/html/latest/security/landlock.html]
- TOMOYO
    - Kernel-Dokumentation: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/tomoyo.rst [https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/LSM/tomoyo.rst]
    - Handbuch: https://tomoyo.osdn.jp/2.6/index.html.en [https://tomoyo.osdn.jp/2.6/index.html.en]
- SMACK
    - SMACK Kernel Dokumentation [https://www.kernel.org/doc/html/v4.14/admin-guide/LSM/Smack.html]
    - Heise iX: SMACK im Linux Kernel [https://www.heise.de/select/ix/2017/3/1488028406202537]
- SELinux
    - NSA SELinux https://web.archive.org/web/20201022103915/https://www.nsa.gov/what-we-do/research/selinux/ [https://web.archive.org/web/20201022103915/https://www.nsa.gov/what-we-do/research/selinux/]
    - Webseite https://selinuxproject.org/page/Main_Page [https://selinuxproject.org/page/Main_Page]
- AppArmor
    - AppArmor Wiki http://wiki.apparmor.net/index.php/Main_Page [http://wiki.apparmor.net/index.php/Main_Page]

Audit Subsystem

- Audit Subsystem Homepage https://people.redhat.com/sgrubb/audit [https://people.redhat.com/sgrubb/audit]
- Linux Audit Mailing-Liste https://listman.redhat.com/mailman/listinfo/linux-audit [https://listman.redhat.com/mailman/listinfo/linux-audit]
- Linux Audit Dokumentation https://github.com/linux-audit/audit-documentation/wiki [https://github.com/linux-audit/audit-documentation/wiki]
- Introduction to Linux Audit http://security-plus-data-science.blogspot.com/2017/02/introduction-to-linux-audit.html [http://security-plus-data-science.blogspot.com/2017/02/introduction-to-linux-audit.html]
- Audit log Normalization Part 1 http://security-plus-data-science.blogspot.com/2017/02/audit-log-normalization.html [http://security-plus-data-science.blogspot.com/2017/02/audit-log-normalization.html]
- Audit log Normalization Part 2 – CSV format http://security-plus-data-science.blogspot.com/2017/02/audit-log-normalization-part-2-csv.html [http://security-plus-data-science.blogspot.com/2017/02/audit-log-normalization-part-2-csv.html]
- A Linux Auditd rule set mapped to MITRE's Attack Framework https://github.com/bfuzzy/auditd-attack [https://github.com/bfuzzy/auditd-attack]
- Setup and configure linux auditd https://github.com/juju4/ansible-auditd [https://github.com/juju4/ansible-auditd]

- Best Practice Auditd Configuration https://github.com/Neo23x0/auditd/ [https://github.com/Neo23x0/auditd/]
- Splunk App for Linux Auditd https://github.com/doksu/splunk_auditd [https://github.com/doksu/splunk_auditd]
- All-Seeing Eye or Blind Man? Understanding the Linux Kernel Auditing System https://www.sans.org/white-papers/38605/ [https://www.sans.org/white-papers/38605/]
- SUSE "Understanding Linux Audit" https://documentation.suse.com/sles/12-SP4/html/SLES-all/cha-audit-comp.html [https://documentation.suse.com/sles/12-SP4/html/SLES-all/cha-audit-comp.html]
- Fedora/Red Hat: Changes/Deprecate TCP wrappers https://fedoraproject.org/wiki/Changes/Deprecate_TCP_wrappers [https://fedoraproject.org/wiki/Changes/Deprecate_TCP_wrappers]
- HOWTO Fedora Enable Auditing https://github.com/linux-audit/audit-documentation/wiki/HOWTO-Fedora-Enable-Auditing [https://github.com/linux-audit/audit-documentation/wiki/HOWTO-Fedora-Enable-Auditing]

SELinux

- SELinux FOR RED HAT DEVELOPERS https://www.redhat.com/en/files/resources/en-rhel-selinux-devlopers-120114.pdf [https://www.redhat.com/en/files/resources/en-rhel-selinux-devlopers-120114.pdf]
- SELinux Notebook https://github.com/SELinuxProject/selinux-notebook [https://github.com/SELinuxProject/selinux-notebook]
- Fedora Security-Enhanced Linux User Guide https://docs.fedoraproject.org/en-US/Fedora/13/html/Security-Enhanced_Linux/index.html [https://docs.fedoraproject.org/en-US/Fedora/13/html/Security-Enhanced_Linux/index.html]
- Fedora SELinux FAQ – Frequently-asked questions about Security Enhanced Linux https://docs.fedoraproject.org/en-US/Fedora/13/html/SELinux_FAQ/index.html [https://docs.fedoraproject.org/en-US/Fedora/13/html/SELinux_FAQ/index.html]
- A Brief Tour of Linux Security Modules https://www.starlab.io/blog/a-brief-tour-of-linux-security-modules/ [https://www.starlab.io/blog/a-brief-tour-of-linux-security-modules/]
- SELinux Struggles with BIND Startup https://www.isc.org/blogs/selinux-struggles-bind/ [https://www.isc.org/blogs/selinux-struggles-bind/]
- `named_selinux` Manual page https://linux.die.net/man/8/named_selinux [https://linux.die.net/man/8/named_selinux] (the man page on your system is likely more up-to-date)
- Policy Quellcode Beispiele https://github.com/SELinuxProject/refpolicy [https://github.com/SELinuxProject/refpolicy]

Bücher

- SELinux System Administration – Third Edition by Sven Vermeulen https://www.packtpub.com/product/selinux-system-administration-third-edition/9781800201477 [https://www.packtpub.com/product/selinux-system-administration-third-edition/9781800201477]
- SELinux & AppArmor: Mandatory Access Control für Linux einsetzen und verwalten von Ralf Spenneberg https://os-s.net/publications/buecher/selinux.pdf [https://os-s.net/publications/buecher/

selinux.pdf] (nicht mehr aktuell)

Anderes

- Instrumenting BIND 9 on Linux with BCC/eBPF
  https://www.youtube.com/watch?v=VYpZg89NJeA [https://www.youtube.com/watch?v=VYpZg89NJeA]

3.3. TCP/IP SICHERHEIT

- TCP/IP Sicherheit [./tcpip-security.org]

3.4. FIREWALL NFTABLES

3.4.1. Einführung in nftables

- Einführung in nftables [./nftables.html]

3.4.2. Beispiel: ein Regelwerk für eine Host-Firewall

- Das nachfolgende *nftable*-Regelwerk kann als Grundlage für eine Host-Firewall für eine Linux-Workstation benutzt werden.
- Gefiltert werden die eingehenden IPv4- und IPv6-Verbindungen.
- Gängige Dienste auf einer Linux-Workstation wie Internet Printing Protocol (IPP/Cups) und Multicast-DNS werden erlaubt:

```
flush ruleset

define any = 0::0/0

table inet filter {
        chain input {
                type filter hook input priority 0;

                # accept any localhost traffic
                iif lo accept

                # accept traffic originated from us
                ct state established,related accept

                # activate the following line to accept common local services
                #tcp dport { 22, 80, 443 } ct state new accept

                # NTP multicast
                ip6 daddr ff02::1010 udp dport 123 accept

                # mDNS (avahi)
                ip6 daddr ff02::fb udp dport 5353 accept
                ip  daddr 224.0.0.251 udp dport 5353 accept

                # DHCPv6
                ip6 saddr $any udp dport 546 accept
```

```
                    # IPP (CUPS)
                    udp dport 631 accept

                    # Accept neighbour discovery otherwise IPv6 connectivity breaks.
                    ip6 saddr $any icmpv6 type { nd-neighbor-solicit,  nd-router-
advert, nd-neighbor-advert }  accept

                    # Accept essential icmpv6
                    # nft cannot parse "param-problem" (icmpv6 type 4)
                    ip6 saddr $any icmpv6 type { echo-reply, echo-request, packet-too-
big, destination-unreachable, time-exceeded, 4 } accept

                    # count and drop any other traffic
                    counter log prefix "nftables drop: " drop
            }
    }
```

3.4.3. Beispiel: ein Regelwerk für einen Gateway-Router mit IPv4-NAT

- Nachfolgend ein *nftables*-Regelwerk für eine einfache Firewall mit DMZ-, WAN- und LAN-Schittstelle. In der DMZ befinden sich die Server mit Web und E-Mail.
- In der Postrouting-Kette werden alle Pakete aus dem LAN-Segment, welche die Firewall in Richtung Internet (WAN) verlassen, per Masquerading-NAT auf die IPv4-Adresse der WAN-Netzwerkschnittelle umgeschrieben.

```
    #!/usr/bin/nft -f
    flush ruleset
    define mgt_if = eth0
    define wan_if = eth1
    define lan_if = eth3
    define dmz_if = eth2
    define any_v6  = 0::0/0

    table inet gateway-fw {
            chain forward {
                    type filter hook forward priority 0
                    # accept established traffic
                    ct state established,related accept
                    # allow all outgoing traffic from LAN
                    iif $lan_if accept
                    # access to services in the DMZ
                    oif $dmz_if tcp dport { http, https, smtp, imap, imaps } counter
accept
                    # Accept essential icmpv6
                    # nft cannot parse "param-problem" (icmpv6 type 4)
                    ip6 saddr $any_v6 icmpv6 type { echo-reply, echo-request, packet-
too-big, destination-unreachable, time-exceeded, 4 } accept
                    # reject everything else
                    counter log reject
            }
            chain input {
                    type filter hook input priority 0
```

```
                    iif lo accept
                    iif $mgt_if tcp dport ssh accept
                    ip6 saddr $any_v6 icmpv6 type { nd-neighbor-solicit,  nd-router-
advert, nd-neighbor-advert }  accept
                    ip6 saddr $any_v6 icmpv6 type { echo-reply, echo-request, packet-
too-big, destination-unreachable, time-exceeded, 4 } accept
                    counter log reject
           }
    }
    table ip nat {
            chain prerouting {
                    type nat hook prerouting priority 0
            }
            chain postrouting {
                    type nat hook postrouting priority 0
                    oif $wan_if log masquerade
            }
    }
```

3.4.4. Aufgabe: IPv4 Host-Firewall

- Stoppe die Firewall-Management-Software `firewalld`. Der Befehl `nft list ruleset` sollte ein leeres *nft* Regelwerk zurückgeben

```
# systemctl stop firewalld
# nft list ruleset
```

- Erstelle eine Host-Firewall in der Datei `/etc/nftables.conf`, welche nur folgende Zugriffe erlaubt
  - ICMP Echo-Request/Echo-Reply
  - Port 80 (HTTP), Port 443 (HTTPS) und 22 (SSH)
- Starte einen Python-Webserver als Benutzer `root` auf Port 80

```
python3 -m http.server 80
```

- Installiere `nmap`

```
dnf install nmap
```

- Scanne per `nmap` nach offenen Ports der VM Deines Nachbarn einmal mit `nftables` Firewall aktiv, einmal ohne eingeschaltete `nftables` Firewall auf der VM des Nachbarn. Gibt es Unterschiede in der Ausgabe?

```
 nmap -v -sT selinuxXXX.linux-sicherheit.org
```

- Teste bei eingeschalteter Firewall auf dem Rechner des Nachbarn, ob die Website/Verzeichnis-Listing auf dem Webserver des Nachbarn vom Browser erreichbar ist.

3.4.5. Erweiterte Aufgabe:

- Starte den Python Webserver auf Port 8000

```
python3 -m http.server 8000
```

- Erstelle eine `redirect` oder `dnat` Destination-NAT Regel (Tabelle `nat`, Chain `prerouting`), welche Zugriffe auf Port 80 des Rechners auf Port 8000 umleitet: *Nftables* Dokumentation: `man nftables`
- Teste dieses Regelwerk mit einem Browser vom einem anderen Rechner, der Browser sollte auf Port 80 den Webserver erreichen können

3.4.6. Beispiel-Lösung

```
#!/usr/sbin/nft -f

flush ruleset

table ip filter {
        chain input {
                type filter hook input priority 0;
        policy accept;

                iif lo accept

        icmp type { echo-reply, echo-request } accept
                ct state established,related accept
                tcp dport { ssh, http, https } ct state new accept

        counter log prefix "nftables drop: " drop
        }
}
```

- Regelwerk nach Änderungen neu laden

```
systemctl restart nftables
```

- aktives Regelwerk inkl. der Zähler anzeigen

```
nft list ruleset
```

- Log-Meldungen erscheinen im Syslog (`/var/log/messages`) und im Journal

```
journalctl | grep nftables
grep nftables /var/log/messages
```

3.4.7. Beispiel-Lösung (Erweitert)

```
#!/usr/sbin/nft -f

flush ruleset
```

```
table ip nat {
    chain prerouting {
        type nat hook prerouting priority 0;
        tcp dport 80 redirect to 8000
    }
    chain postrouting {
        type nat hook postrouting priority 0;
    }

}

table ip filter {
    chain input {
        type filter hook input priority 0;
        policy accept;

        iif lo accept

        icmp type { echo-reply, echo-request } accept
        ct state established,related accept
        tcp dport { ssh, http, https, 8000 } ct state new accept

        counter log prefix "nftables drop: " drop
    }
}
```

3.5. LOGDATEN ÜBERWACHEN

3.5.1. Artificial Ignorance

- Bekannte *negative* oder *bösartige* Log-Meldung verfolgen und abstellen
- Bekannt *gutartige* Log-Meldungen (welche nicht abgestellt werden können) ausfiltern (granular, nicht *überfiltern*, Vorsicht bei RegExps)

3.5.2. Quellen von sicherheitsrelevanten Log-Informationen

- Liste von Log-Informationen aus RHEL Systemen welche in SIEM-Systemen überwacht werden sollten
    - Kernel-Log (via Systemd-Journal)
        - Firewall
        - Kernel Modul Laden/Entladen
        - Hardware Fehler
        - Abnormale Prozess-Zustände (oomkiller, crashes)
        - Kernel-Panics
        - Kernel-Updates (Geänderte Kernel-Versionen)
        - Dateisystem Mounts

        - Uhrzeit Änderungen/Drifts
- Audit-Log (via `auditd`)
    - SELinux AVC Events
    - Zugriffe auf sensitive Daten (private Schlüssel, DNS-Resolver Konfiguration, Kernel-Dateien, SUID-Bit Dateien etc) – benötigt eine Audit-Subsystem Regel-Datei
    - Login-Informationen (Benutzer-Anmeldungen/Anmelde-Versuche)
    - Rechte-Erweiterungen (`su=/=sudo`)
- Systemd-Journal
    - Dienst `systemd-journald`
    - Dienst `chronyd`
    - DHCPv4/DHCPv6 Events (NetworkManager, systemd-networkd)
    - Dienst `auditd`
    - Dienst `kdump`
    - Dienst `ldconfig`
    - Dienst `sshd`
    - Dienst `systemd-udevd`

3.6. NETZWERK-DATENLECKS

- Datenlecks durch Fehlkonfigurationen
    - Sensible Daten nicht auf Internet-Servern (Server welche direkt aus dem Internet erreichbar sind oder im Internet gehostet werden, inkl. Cloud-Dienste) speichern oder verarbeiten – Darstellung und Verarbeitung trennen (z.B. über API zu Backend-Servern)
    - Generische DNS-Namen verwenden, keine "Produkt"-Namen
        - Schlecht:
            - `fortigate.firma.de`
            - `nextcloud.firma.de`
            - `ansible-tower01.firma.de`
        - Besser:
            - `ipgateway.firma.de`
            - `cloud.firma.de`
            - `cnfmgmt01.firma.de`
    - Vorsicht bei Tunnel-Setup (z.B. TOR) auf Loopback-Adressen (127.0.0.1). Einige Software-Systeme geben automatisch erweiterte Rechte auf Verbindungen von Loopback-Adressen
- Datenlecks durch Angriffe
    - Robuste Software-Architekturen erstellen/von Dienstleistern einfordern
        - Sensible Daten via *push* auf Internet-Server liefern (Datenverbindung nur von internen

Netzen in externe Netze), z.B. für Updates von TLS-Zertifikaten

- Daten auf Servern im Internet nur im Hauptspeicher halten
  - Keine temporären Dateien anlegen
  - Keine lokalen Datenbanken auf Frontend-Systemen
  - Keine-Swap/Paging-Dateien – ZRam als Swap benutzen

- DNS / ICMP / HTTPS / QUICK
  - Daten können über viele Protokolle exfiltriert werden
    - Payload in ICMP-Echo Paketen
    - DNS-over-TLS/DNS-over-HTTPS/DNS-Tunnel (TCP/IP over DNS)
    - HTTPS/TLS 1.3 kann nicht (einfach) aufgebrochen werden
    - HTTP/3 (aka QUICK) Implementiert TLS 1.3 direkt im Basis-Protokoll, mehrere *Streams* in einer Sitzung
    - Datenverkehr-überwachung muss ggf. auf die Hosts (Server, Laptop, Desktop) Systeme umziehen, zentrale Überwachung vom Datenverkehr am Internet-Gateway nicht mehr zuverlässig möglich

3.7. ROOTKITS AUFSPÜREN

- Root-Kit-Hunter installieren

```
dnf install rkhunter
```

- Signatur-Updates einspielen

```
rkhunter --update
```

- RKHunter Baseline Datenbank erstellen

```
rkhunter --propupd
```

- nach Root-Kits suchen

```
rkhunter --check
less /var/log/rkhunter.log
```

- Informationen zu Root-Kits unter Linux
  - http://la-samhna.de/library/rootkits/list.html [http://la-samhna.de/library/rootkits/list.html]
  - https://www.sans.org/reading-room/whitepapers/linux/linux-rootkits-beginners-prevention-removal-901 [https://www.sans.org/reading-room/whitepapers/linux/linux-rootkits-beginners-prevention-removal-901]
  - https://lwn.net/Articles/224140/ [https://lwn.net/Articles/224140/]

3.8. RHASH

- rhash Installation

```
dnf 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
```

3.9. SOFTWARE-SICHERHEITSFUNKTIONEN UNTER LINUX

3.9.1. 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://capsule8.com/blog/millions-of-binaries-later-a-look-into-linux-hardening-in-the-wild/ [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]

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

3.10. LINUX-DISTRIBUTIONEN (SICHERHEITSBEWERTUNG)

- Debian 11/12
  - 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 / CentOS / Rocky / Alma / 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: ++

3.11. LINUX SICHERHEITSMELDUNGEN

Quelle	Link
Linux Weekly News Sicherheitsmeldungen	https://lwn.net/Security/ [https://lwn.net/Security/]
Red Hat Sicherheitsmeldungen	<a href="https://access.redhat.com/security/security–updates/<mark>/security–advisories">https://access.redhat.com/security/security–updates/</mark>/security–advisories [https://access.redhat.com/security/security–updates/<mark>/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]

Quelle	Link
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]