Sicherheit

Firewall & Co.

- Linux mit ufw (Uncomplicated FireWall) absichern
- Upgrade-Script für Debian / Raspi OS
- Absichern des SSH-Logins mit 2-Faktor-Authentifizierung (TOTP)
- Linux - Watchdog
- Linux - Kernel Parameter (sysctl)

Linux mit ufw (Uncomplicated FireWall) absichern

Installation

UFW installieren:

```
apt install ufw
```

IPv6 bei Bedarf deaktivieren

Editieren der Datei /etc/default/ufw:

```
nano /etc/default/ufw
```

Um IPv6 zu deaktivieren, ändern der Zeile IPV6=yes auf IPV6=no:

```
# /etc/default/ufw
#

# Set to yes to apply rules to support IPv6 (no means only IPv6 on loopback
# accepted). You will need to 'disable' and then 'enable' the firewall for
# the changes to take affect.
IPV6=no
```

Aufrufen der Hilfe

```
ufw help
```

```
Usage: ufw COMMAND

Commands:
 enable                          enables the firewall
 disable                         disables the firewall
 default ARG                     set default policy
 logging LEVEL                   set logging to LEVEL
 allow ARGS                      add allow rule
 deny ARGS                       add deny rule
```

```
 reject ARGS                    add reject rule
 limit ARGS                     add limit rule
 delete RULE|NUM                delete RULE
 insert NUM RULE                insert RULE at NUM
 route RULE                     add route RULE
 route delete RULE|NUM          delete route RULE
 route insert NUM RULE          insert route RULE at NUM
 reload                         reload firewall
 reset                          reset firewall
 status                         show firewall status
 status numbered                show firewall status as numbered list of RULES
 status verbose                 show verbose firewall status
 show ARG                       show firewall report
 version                        display version information

Application profile commands:
 app list                       list application profiles
 app info PROFILE               show information on PROFILE
 app update PROFILE             update PROFILE
 app default ARG                set default application policy
```

Vorgefertigte Profile

Die ufw hat vorgefertigte Profile für verschiedene Dienste:

```
ufw app list
```

```
Available applications:
  AIM
  Bonjour
  CIFS
  DNS
  Deluge
  IMAP
  IMAPS
  IPP
  KTorrent
  Kerberos Admin
  Kerberos Full
  Kerberos KDC
  Kerberos Password
```

```
  LDAP
  LDAPS
  LPD
  MSN
  MSN SSL
  Mail submission
  NFS
  OpenSSH
  POP3
  POP3S
  PeopleNearby
  SMTP
  SSH
  Socks
  Telnet
  Transmission
  Transparent Proxy
  VNC
  WWW
  WWW Cache
  WWW Full
  WWW Secure
  XMPP
  Yahoo
  qBittorrent
  svnserve
```

Die Dienste verwenden die Standardports, so z.B. Port 22 für SSH, Port 80 für WWW oder Port 443 für WWW Secure.

Weitere Informationen bezüglich der Ports finden sich in den Konfigurationsdateien im Verzeichnis /etc/ufw/applications.d:

```
ls -al /etc/ufw/applications.d
```

```
insgesamt 52
drwxr-xr-x 2 root root 4096 Jun  7 13:58 .
drwxr-xr-x 3 root root 4096 Feb  7  2018 ..
-rw-r--r-- 1 root root  145 Nov 18  2017 openssh-server
-rw-r--r-- 1 root root  353 Feb 18  2016 ufw-bittorent
-rw-r--r-- 1 root root  627 Feb 18  2016 ufw-chat
```

```
-rw-r--r-- 1 root root  513 Feb 18  2016 ufw-directoryserver
-rw-r--r-- 1 root root   89 Feb 18  2016 ufw-dnsserver
-rw-r--r-- 1 root root  358 Feb 18  2016 ufw-fileserver
-rw-r--r-- 1 root root  212 Feb 18  2016 ufw-loginserver
-rw-r--r-- 1 root root  524 Feb 18  2016 ufw-mailserver
-rw-r--r-- 1 root root  131 Feb 18  2016 ufw-printserver
-rw-r--r-- 1 root root  155 Feb 18  2016 ufw-proxyserver
-rw-r--r-- 1 root root  320 Feb 18  2016 ufw-webserver
```

```
cat ufw-webserver
```

```
[WWW]
title=Web Server
description=Web server
ports=80/tcp

[WWW Secure]
title=Web Server (HTTPS)
description=Web Server (HTTPS)
ports=443/tcp

[WWW Full]
title=Web Server (HTTP,HTTPS)
description=Web Server (HTTP,HTTPS)
ports=80,443/tcp

[WWW Cache]
title=Web Server (8080)
description=Web Server (8080)
ports=8080/tcp
```

Die Konfigurationsdateien in diesem Verzeichnis können durch eigene Einträge ergänzt oder es können auch neue Konfigurationsdateien erstellt werden.

Regeln hinzufügen

In diesem Beispiel wird der SSH-Zugang für das lokale Netzwerk (192.168.1.0/24) und ein Webzugriff für jede IP eingerichtet, IPv6 wurde deaktiviert:

```
ufw allow from 192.168.1.0/24 to any app SSH
```

```
ufw allow 'WWW Full'
```

Die Firewall starten:

```
ufw enable
```

Die Durchführung mit ‚y' bestätigen:

```
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```

Status abfragen

```
ufw status
```

```
Status: active

To                         Action      From
--                         ------      ----
SSH                        ALLOW       192.168.1.0/24
WWW Full                   ALLOW       Anywhere
```

Einen detailierten Status mit Angabe der Ports ausgeben:

```
ufw status verbose
```

```
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp (SSH)               ALLOW IN    192.168.1.0/24
80,443/tcp (WWW Full)      ALLOW IN    Anywhere
```

Löschen von Regeln

Ausgabe des Status mit Nummern:

```
ufw status numbered
```

```
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] SSH                        ALLOW IN    192.168.1.0/24
[ 2] WWW Full                   ALLOW IN    Anywhere
```

Löschen der Regel über die Nummer:

```
ufw delete 2
```

Beispiele

Eingehende Pakete auf Port 12345 zulassen:

```
ufw allow from 192.168.1.0/24 to any port 12345
```

Eingehende UDP Pakete auf Port 54321 zulassen:

```
ufw allow from 192.168.1.0/24 proto udp to any port 54321
```

Eine neue Regel an erster Postition einfügen:

```
ufw prepend allow from ...
```

Eine neue Regel an der Position 2 einfügen:

```
ufw insert 2 allow from ...
```

> Standardmäßig werden ausgehende Pakete immer akzeptiert. Um dies zu ändern muss die Einstellung in der **/etc/default/ufw** auf **DEFAULT_OUTPUT_POLICY="DROP"** geändert werden.

Ausgehende Pakete zu 192.168.1.1 blocken:

```
ufw deny out from any to 192.168.1.1
```

Ausgehende DNS Anfragen (Port 123/UDP) an 192.168.1.1 zulassen:

```
ufw allow out to 192.168.1.1 port 123 proto udp
```

Upgrade-Script für Debian / Raspi OS

> Ein Script zum Updaten von Debian und Raspi OS

Abhängigkeiten

Das Script ruft das Tool **needrestart** auf:

```
apt install needrestart
```

Installation

Alles in die Shell kopieren um die Datei **debianupdate.sh** zu erstellen:

```
cat > /root/debianupdate.sh << EOF

#!/bin/bash

lightblue='\033[1;34m'
lightgreen='\033[1;32m'
lightred='\033[1;31m'
yellow='\033[0;33m'
nocolor='\033[0m'

echo -e "\n\${lightblue}Run command 'apt-get update':\${nocolor}"
apt-get update
echo -e "\${lightgreen}Command 'apt-get update' finished.\${nocolor}"

echo -e "\n\${lightblue}Run command 'apt-get dist-upgrade':\${nocolor}"
apt-get dist-upgrade
echo -e "\${lightgreen}Command 'apt-get dist-upgrade' finished.\${nocolor}"

echo -e "\n\${lightblue}Run command 'apt-get autoremove':\${nocolor}"
apt-get autoremove
echo -e "\${lightgreen}Command 'apt-get autoremove' finished.\${nocolor}"
```

```
echo -e "\n\${lightblue}Run command 'apt-get autoclean':\${nocolor}"
apt-get autoclean
echo -e "\${lightgreen}Command 'apt-get autoclean' finished.\${nocolor}"

echo -e "\n\${lightblue}Run command 'needrestart' to check which processes need to be restarted
after an upgrade:\${nocolor}"
if test -e /usr/sbin/needrestart
then
    needrestart
else
    echo -e "\n\${yellow}Command 'needrestart' not found, use 'apt install needrestart' to
install this feature!\${nocolor}"
fi
echo -e "\n\${lightgreen}Update now completed.\${nocolor}"
exit 0
EOF
```

Berechtigungen

Die Berechtigungen ändern:

```
chmod 500 /root/debianupdate.sh
```

Ausführen

Ausführen des Upgrades mit:

```
./debianupdate.sh
```

Absichern des SSH-Logins mit 2-Faktor-Authentifizierung (TOTP)

Installation

Installieren des **libpam-google-authenticator** Pakets:

```
apt install libpam-google-authenticator
```

Einrichtung

> Die Erstellung des Geheimschlüssels und der Notfallscodes erfolgt immer für den aktuell angemeldeten Benutzer, in dieser Anleitung für den Benutzer root.

Starten des **google-authenticator**:

```
google-authenticator
```

Die 1. Frage, ob mit Zeitbasierten Token (TOTP) gearbeitet werden soll, mit **y** beantworten:

```
Do you want authentication tokens to be time-based (y/n)
```

> Die angezeigete URL **NICHT** per Browser aufrufen, da sonst der Geheimschlüssel an Google übertragen wird!

> Den QR-Code mit den gewünschten Geräten scannen und den Geiheimschlüssel im Passwortsafe hinterlegen.

Die 2. Frage, ob die Datei **/root/.google_authenticator** aktualisiert werden soll, mit **y** beantworten:

```
Do you want me to update your "/root/.google_authenticator" file? (y/n)
```

Die 3. Frage, ob auf einen Login alle 30 Sekunden beschränkt werden sollen, mit **y** beantworten:

```
Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n)
```

Die 4. Frage, ob von 3 auf 17 erlaubte Codes erweitert werden soll, mit **n** beantworten:

```
By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If you
experience problems with poor time synchronization, you can increase the window
from its default size of 3 permitted codes (one previous code, the current
code, the next code) to 17 permitted codes (the 8 previous codes, the current
code, and the 8 next codes). This will permit for a time skew of up to 4 minutes
between client and server.
Do you want to do so? (y/n)
```

Die 5. Frage, ob auf maximal 3 Loginversuche pro 30 Sekunden beschränkt werden soll, mit **y** beantworten:

```
If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting? (y/n)
```

Änderungen an der /etc/pam.d/sshd

Am Beginn der Datei /etc/pam.d/sshd die Zeile **auth required pam_google_authenticator.so** hinzufügen:

```
patch -u /etc/pam.d/sshd << EOF
@@ -1,4 +1,5 @@
 # PAM configuration for the Secure Shell service
+auth required pam_google_authenticator.so

 # Standard Un*x authentication.
 @include common-auth
EOF
```

Änderungen an der /etc/ssh/sshd_config

In der **/etc/ssh/sshd_config** den Schlüssel auf **KbdInteractiveAuthentication yes** ändern:

```
sed -i 's/ChallengeResponseAuthentication no/KbdInteractiveAuthentication yes/g'
/etc/ssh/sshd_config
sed -i 's/KbdInteractiveAuthentication no/KbdInteractiveAuthentication yes/g'
/etc/ssh/sshd_config
```

> In älteren Installationen wird noch **ChallengeResponseAuthentication** verwendet, dieses wurde durch **KbdInteractiveAuthentication** ersetzt.

Den ssh-Server neu starten:

```
systemctl restart sshd
```

Linux - Watchdog

> Anleitung zur Installation und Konfiguration des Pakets **watchdog** (Link zur Projektseite).

> Die Anleitungen zum Aktivieren der Watchdogs für **Proxmox** und **Raspberry Pi** finden sich hier:
>
> Virtuelles Watchdog-Modul für Proxmox
>
> Hardware-Watchdog für Raspberry Pi

Installieren des Watchdog Pakets

```
apt install watchdog
```

Konfigurieren des Watchdogs

Folgende Änderungen in der Datei **/etc/watchdog** durchführen:

```
watchdog-device         = /dev/watchdog
watchdog-timeout        = 15
```

Testen des Watchdogs

> Der folgende Befehl triggert eine Kernel Panic worauf der Rechner neu bootet.

```
echo c > /proc/sysrq-trigger
```

Beispiele

> Folgende Beispiele lassen den Watchdog einen **reboot** durchführen **wenn die Bedingung NICHT zutrifft**.

Interface überträgt Daten:

```
interface               = eth0
```

IP-Adresse pingbar:

```
ping                    = 172.16.0.1
```

Max. Load ist innerhalb 1 Minute unter 24%, innerhalb 5 Minuten unter 18%, innerhalb 15 Minuten unter 12%:

```
max-load-1              = 24
max-load-5              = 18
max-load-15             = 12
```

Skripte

Skripte können in das Verzeichnis **/etc/watchdog.d** (Vorgabe in der **watchdog.conf**) gelegt werden und werden Automatisch vom Watchdog zum Testen verwendet. Der Watchdog wird bei einem **exit 0** den Watchdog-Timer zurücksetzen. Bei einem **exit 255** (-1) wird ein reboot durchgeführt.

```
#!/bin/bash

# Es wird ein Befehl ausgeführt und die Ausgabe in die Variable geschrieben
# Hier wird z.B. nach einen USB-Gerät gesucht, bei welchem in der Zeile "Titanium" vorkommt
listusb=`lsusb | grep Titanium`

if [[ -z  $listusb ]]; then # Wenn die Variable leer ist (Gerät nicht vorhanden)
        exit 255 # Rückgabewert -1 Neustart (reboot)
fi

exit 0 # Wert 0 setzt Watchdog Timer zurück
```

> Als Rückgabewerte (exit codes) für den Watchdog stehen folgende Werte zu Verfügung:
>
> 0 (exit 0): Kein Fehler - Watchdog wird zurückgesetzt
> -1 (exit 255): Neustart (reboot)
> -2 (exit 254): Zurücksetzen (reset)
> -3 (exit 253): Maximale durchschnittliche Auslastung überschritten.
> -4 (exit 252): Die Innentemperatur ist zu hoch.
> -5 (exit 251): /proc/loadavg enthält keine (oder nicht genügend) Daten.
> -6 (exit 250): Die angegebene Datei wurde im angegebenen Intervall nicht geändert.
> -7 (exit 249): /proc/meminfo enthält ungültige Daten.
> -8 (exit 248): Der Kindprozess wurde durch ein Signal beendet.
> -9 (exit 247): Der Kindprozess ist nicht rechtzeitig zurückgekehrt.
> -10 (exit 246): Für den persönlichen Gebrauch frei.

/etc/watchdog.conf (Beispieldatei)

```
# ======================================================================
# Configuration for the watchdog daemon. For more information on the
# parameters in this file use the command 'man watchdog.conf'
# ======================================================================

# ================== The hardware timer settings ====================
#
# For this daemon to be effective it really needs some hardware timer
# to back up any reboot actions. If you have a server then see if it
# has IPMI support. Otherwise for Intel-based machines try the iTCO_wdt
# module, otherwise (or if that fails) then see if any of the following
# module load and work:
#
# it87_wdt it8712f_wdt w83627hf_wdt w83877f_wdt w83977f_wdt
#
# If all else fails then 'softdog' is better than no timer at all!
# Or work your way through the modules listed under:
#
# /lib/modules/`uname -r`/kernel/drivers/watchdog/
#
# To see if they load, present /dev/watchdog, and are capable of
# resetting the system on time-out.

# Uncomment this to use the watchdog device driver access "file".

watchdog-device         = /dev/watchdog

# Uncomment and edit this line for hardware timeout values that differ
# from the default of one minute.

watchdog-timeout        = 15

# If your watchdog trips by itself when the first timeout interval
# elapses then try uncommenting the line below and changing the
# value to 'yes'.

#watchdog-refresh-use-settimeout        = auto
```

```
# If you have a buggy watchdog device (e.g. some IPMI implementations)
# try uncommenting this line and setting it to 'yes'.

#watchdog-refresh-ignore-errors = no

# ====================== Other system settings =======================
#
# Interval between tests. Should be a couple of seconds shorter than
# the hardware time-out value.

#interval               = 1

# The number of intervals skipped before a log message is written (i.e.
# a multiplier for 'interval' in terms of syslog messages)

#logtick        = 1

# Directory for log files (probably best not to change this)

#log-dir                = /var/log/watchdog

# Email address for sending the reboot reason. This needs sendmail to
# be installed and properly configured. Maybe you should just enable
# syslog forwarding instead?

#admin                  = root

# Lock the daemon in to memory as a real-time process. This greatly
# decreases the chance that watchdog won't be scheduled before your
# machine is really loaded.

realtime                = yes
priority                = 1

# ====================== How to handle errors  =======================
#
# If you have a custom binary/script to handle errors then uncomment
# this line and provide the path. For 'v1' test binary files they also
# handle error cases.
```

```
#repair-binary          = /usr/sbin/repair
#repair-timeout         = 60

# The retry-timeout and repair limit are used to handle errors in a
# more robust manner. Errors must persist for longer than this to
# action a repair or reboot, and if repair-maximum attempts are
# made without the test passing a reboot is initiated anyway.

#retry-timeout          = 60
#repair-maximum         = 1

# Configure the delay on reboot from sending SIGTERM to all processes
# and to following up with SIGKILL for any that are ignoring the polite
# request to stop.

#sigterm-delay          = 5

# ====================== User-specified tests ========================
#
# Specify the directory for auto-added 'v1' test programs (any executable
# found in the 'test-directory should be listed).

#test-directory = /etc/watchdog.d

# Specify any v0 custom tests here. Multiple lines are permitted, but
# having any 'v1' programs/scripts discovered in the 'test-directory' is
# the better way.

#test-binary            =

# Specify the time-out value for a test error to be reported.

#test-timeout           = 60

# ====================== Typical tests ================================
#
# Specify any IPv4 numeric addresses to be probed.
# NOTE: You should check you have permission to ping any machine before
# using it as a test. Also remember if the target goes down then this
```

```
# machine will reboot as a result!

#ping                   = 172.16.0.1
#ping                   = 192.168.1.1

# Set the number of ping attempts in each 'interval' of time. Default
# is 3 and it completes on the first successful ping.
# NOTE: Round-trip delay has to be less than 'interval' / 'ping-count'
# for test success, but this is unlikely to be exceeded except possibly
# on satellite links (very unlikely case!).

#ping-count             = 3

# Specify any network interface to be checked for activity.

#interface              = eth0

# Specify any files to be checked for presence, and if desired, checked
# that they have been updated more recently than 'change' seconds.

#file                   = /var/log/syslog
#change                 = 1407

# Uncomment to enable load average tests for 1, 5 and 15 minute
# averages. Setting one of these values to '0' disables it. These
# values will hopefully never reboot your machine during normal use
# (if your machine is really hung, the loadavg will go much higher
# than 25 in most cases).

#max-load-1             = 24
#max-load-5             = 18
#max-load-15            = 12

# Check available memory on the machine.
#
# The min-memory check is a passive test from reading the file
# /proc/meminfo and computed from MemFree + Buffers + Cached
# If this is below a few tens of MB you are likely to have problems.
#
# The allocatable-memory is an active test checking it can be paged
```

```
# in to use.
#
# Maximum swap should be based on normal use, probably a large part of
# available swap but paging 1GB of swap can take tens of seconds.
#
# NOTE: This is the number of pages, to get the real size, check how
# large the pagesize is on your machine (typically 4kB for x86 hardware).

#min-memory             = 1
#allocatable-memory     = 1
#max-swap = 0

# Check for over-temperature. Typically the temperature-sensor is a
# 'virtual file' under /sys and it contains the temperature in
# milli-Celsius. Usually these are generated by the 'sensors' package,
# but take care as device enumeration may not be fixed.

#temperature-sensor     =
#max-temperature        = 90

# Check for a running process/daemon by its PID file. For example,
# check if rsyslogd is still running by enabling the following line:

#pidfile                = /var/run/rsyslogd.pid
```

Linux - Kernel Parameter (sysctl)

> Der Befehl **sysctl** wird in Linux verwendet um Kernel-Parameter zur Laufzeit auszulesen und zu ändern.

Auflisten aller Kernel Parameter

```
sysctl -a
```

Auslesen eines Kernel Parameters

```
sysctl net.ipv6.conf.all.disable_ipv6
```

Temporäres ändern eines Kernel Parameters

```
sysctl net.ipv6.conf.all.disable_ipv6 = 1
```

Permanentes ändern von kernel Parametern

Um Kernel Parameter permanent zu ändern legt man eine Datei mit den gewünschten Kernel Parametern im Verzeichnis **/etc/sysctl.d** an:

```
nano /etc/sysctl.d/100-mykernelparameters.conf
```

> Die Zahl (Prefix) gibt die Reihenfolge der Ausführung an. Ein Wert in einer Datei mit niedrigem Prefix (z. B. 10) würde durch den Wert in einer Datei mit höherem Prefix (z. B. 100) überschrieben werden!

Beispiel

```
# Kernel security settings
dev.tty.ldisc_autoload = 0
fs.protected_fifos = 2
kernel.core_uses_pid = 1
kernel.dmesg_restrict = 1
kernel.kexec_load_disabled = 1
kernel.kptr_restrict = 2
kernel.randomize_va_space = 2
kernel.sysrq = 0
```

```
kernel.unprivileged_bpf_disabled = 1
kernel.yama.ptrace_scope = 1
net.core.bpf_jit_harden = 2

# IPv4 settings
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.log_martians = 1
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_syncookies = 1

# disable IPv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1

# disable IPv6 redirects
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
```