Linux / Raspberry Pi

Rasberry Pi Sachen, Linux Betriebssysteme etc...

- Automatisierte Speedtests zur Durchklage beim ISP
- AdGuard Home
- Raspberry Pi LEDs abschalten
- Internet-Monitoring auf RPI

Automatisierte Speedtests zur Durchklage beim ISP

> Denkt dran. Es gibt vertragliche Minimums und Maximums. Das Minimum muss konstant, drei Monate lang, zuverlässig und zu verschiedenen Zielen unterschritten worden sein. Das ganze muss direkt an eurem Router hängen (oder es muss zusätzlich nachgewiesen werden dass jede Zwischenkomponente zu jeder Zeit die Minimumbandbreite gekonnt hat).

Mit Grafana

https://pimylifeup.com/raspberry-pi-internet-speed-monitor

Führt bitte nichts aus ohne zu verstehen was es ist. Ich fasse hier ganze HowTo's als ein einzelnes Script zusammen. Wer das in Menschensprache verstehen will muss schon auf den Link klicken.

Diese Scripte sind nicht unbedingt als solche ausführbar! Sie zeigen halt nur ohne Plemplem um Blabla das was an Maschineninput gemacht werden muss.

```
sudo apt-get update && sudo apt-get upgrade -y
sudo apt install apt-transport-https gnupg1 dirmngr lsb-release
curl -L https://packagecloud.io/ookla/speedtest-cli/gpgkey | gpg --dearmor | sudo tee
/usr/share/keyrings/speedtestcli-archive-keyring.gpg >/dev/null
echo "deb [signed-by=/usr/share/keyrings/speedtestcli-archive-keyring.gpg]
https://packagecloud.io/ookla/speedtest-cli/debian/ $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/speedtest.list
sudo apt update
sudo apt install speedtest
curl https://repos.influxdata.com/influxdata-archive.key | gpg --dearmor | sudo tee
/usr/share/keyrings/influxdb-archive-keyring.gpg >/dev/null
echo "deb [signed-by=/usr/share/keyrings/influxdb-archive-keyring.gpg]
https://repos.influxdata.com/debian $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/influxdb.list
sudo apt update
sudo apt install influxdb
```

```
sudo systemctl unmask influxdb
sudo systemctl enable influxdb
sudo systemctl start influxdb
# careful, I don't know yet if setting an admin before activating http auth is even possible.
Create the admin after influxdb.conf edit if it isn't possible
influx
CREATE USER "admin" WITH PASSWORD '<password>' WITH ALL PRIVILEGES
CREATE DATABASE <wie auch immer eure Speedtest DB heißen soll>
USE <wie auch immer eure Speedtest DB heißen soll>
CREATE USER "Speedtest-user" WITH PASSWORD '<password>'
GRANT ALL ON "<wie auch immer eure Speedtest DB heißen soll>" to "Speedtest-user"
quit
sudo sed -i '/^\[HTTP\]/a auth-enabled = true\npprof-enabled = true\npprof-auth-enabled =
true\nping-auth-enabled = true' /etc/influxdb/influxdb.conf
sudo systemctl restart influxdb
sudo apt install python3-influxdb
curl https://apt.grafana.com/gpg.key | gpg --dearmor | sudo tee /usr/share/keyrings/grafana-
archive-keyrings.gpg >/dev/null
echo "deb [signed-by=/usr/share/keyrings/grafana-archive-keyrings.gpg] https://apt.grafana.com
stable main" | sudo tee /etc/apt/sources.list.d/grafana.list
sudo apt update
sudo apt install grafana
sudo systemctl enable grafana-server
sudo systemctl start grafana-server
echo "Grafana is http://$(hostname -I):3000"
echo "influx is http://localhost:8086"
(crontab -l ; echo "*/30 * * * * python3 $(getent passwd \$(whoami) | cut -d: -
f6)/speedtest.py") | crontab -
# replace the variables for the db user before pasting!
cat <<EOF > ~/speedtest.py
import re
import subprocess
from influxdb import InfluxDBClient
host_name = socket.gethostname()
response = subprocess.Popen('/usr/bin/speedtest --accept-license --accept-gdpr', shell=True,
stdout=subprocess.PIPE).stdout.read().decode('utf-8')
ping = re.search('Latency:\s+(.*?)\s', response, re.MULTILINE)
download = re.search('Download:\s+(.*?)\s', response, re.MULTILINE)
upload = re.search('Upload:\s+(.*?)\s', response, re.MULTILINE)
jitter = re.search('Latency:.*?jitter:\s+(.*?)ms', response, re.MULTILINE)
```

```
ping = ping.group(1)
download = download.group(1)
upload = upload.group(1)
jitter = jitter.group(1)
speed_data = [
    {
        "measurement" : "internet_speed",
        "tags" : {
            "host": host_name
        },
        "fields" : {
            "download": float(download),
            "upload": float(upload),
            "ping": float(ping),
            "jitter": float(jitter)
        }
    }
]
client = InfluxDBClient('localhost', 8086, 'Speedtest-user', '<password>', '<wie auch immer
eure Speedtest DB heißen soll>')
client.write_points(speed_data)
EOF
```

Ich habe hier nicht /home/pi/speedtest.py genommen, weil man mit dem RPI imager entsprechend schon User vordefinieren kann. Ich wollte hier eine Lösung haben, die auch für diejenigen gültig ist die das getan haben. Dann gibts nämlich keinen pi user.

Derjenige der diesen Crontab anlegt wird der Script-Ausführer sein.

Spätestens jetzt läuft Grafana und Speedtests werden alle 30 Minuten ins Grafana geschrieben. Webinterface müsste Port 3000 sein. Was ihr hier in Zeile 15 gemacht habt müssten dort die Logindaten sein.

Unschöner mit Gnuplot

Dreckig interpretiert von https://supportblog.ch/uberwache-deine-internetspeedtest-cli/ (weil ich nicht erwarte, dass die Seite länger lebt als mein Eintrag)

Das hier ist erst mal nur interpretiert, daher ungetestet:

```
sudo apt-get install curl
curl -s https://install.speedtest.net/app/cli/install.deb.sh | sudo bash
sudo apt-get install speedtest
speedtest-cli -f csv --output-header >>/var/tmp/speedtests.csv
sed -i '"date time",' /var/tmp/speedtests.csv
sudo crontab -l > cron_bkp
sudo echo "22 */4 * * * /bin/bash -c 'echo "\"$(date +\%d.\%m.\%Y\
\%H:\%M:\%S)\",$(/usr/bin/speedtest -f csv)" >>/var/tmp/speedtests.csv'" >> cron_bkp
sudo crontab cron_bkp
sudo rm cron_bkp
```

Vorsicht Zeile 7 - ist meine unsicherste. Sonst crontab -e verwenden.

Das ganze Gnuplot-Gedöns habe ich hier mal weggelassen.
Backup vom Script mach ich aber mal.

```
cat gnuplot_speedtest_data
# Gnuplot for speedtests.csv
reset
set terminal dumb ansirgb feed enhanced size 128, 31 aspect 1;
set tics nomirror scale 0.5;
set autoscale;
set datafile separator ","

set title "Internetgeschwindigkeiten in Mbps\n[".strftime("%d.%m.%Y %H:%M:%S",time(0))."]"

set xlabel "Datetime" offset 0,-1  # label for the X axis
set xdata time
set timefmt "%d.%m.%Y %H:%M:%S"
set format x "%d.%m\n%H:%M" # Display date time format on graph
# Show last 48h
set xrange [time(0)-2*24*60*60:]

set ylabel "Mbps" offset character 2,0 # label for the Y axis
#set yrange [0:] # show whole range from zero

# change legend position
set key right below nobox

set grid
```

```
plot "/var/tmp/speedtests.csv" using 1:($7*8/1000/1000) with line title "Download", '' using 1:($8*8/1000/1000) with line title "Upload"
```

Aufruf:

```
gnuplot -p gnuplot_speedtest_data
```

AdGuard Home

> Adugard Home erstetzt KEINE Adblocker-Erweiterungen, da keine Websiteinhalte verhindert werden können. Er nimmt diesem aber sehr viel Arbeit ab. Siehe Abschnitt "Einschrönkungen"

Installation

Bitte erst verstehen was hier gemacht wird. Dies kommt aus meiner Hand und erzwingt Downloadmethoden falls keine passenden Programme installiert sind. Befehle im Wiki von Adguard oder unten mit wget.

```
url="https://raw.githubusercontent.com/AdguardTeam/AdGuardHome/master/scripts/install.sh"
download_commands=("curl -s -S -L" "wget --no-verbose -O -" "fetch -o -")
installed=()

if [ "$EUID" -ne 0 ]
  then echo "Please run as root or use sudo"
  exit
fi



for cmd in "${download_commands[@]}"; do
    if command -v ${cmd%% *} >/dev/null 2>&1; then
        echo "using ${cmd%% *}..."
        ${cmd} $url | sh -s -- -v && exit
    else
        echo "${cmd%% *} not found. Attempting to install ${cmd%% *}..."
        sudo apt-get update && sudo apt-get install -y ${cmd%% *}
        if command -v ${cmd%% *} >/dev/null 2>&1; then
            echo "${cmd%% *} installed successfully. Using ${cmd%% *} for installation..."
            ${cmd} $url | sh -s -- -v && exit
            installed+=(${cmd%% *})
        else
            echo "Error: Failed to install ${cmd%% *}. Please install a suitable download
method (curl, wget or fetch) manually and try again..."
```

```
            exit 1
        fi
    fi
done

sudo ./AdGuardHome -s install
sudo ./AdGuardHome -s start

# Uninstall the installed download tools
for cmd in "${installed[@]}"; do
    sudo apt-get remove -y ${cmd}
done
```

Danach Webinterface auf :3000 durchklicken und Router-DNS auf die IP der Maschine setzen

Weniger komplex, dafür kein durchlaufendes script mehr:

```
wget --no-verbose -O -
https://raw.githubusercontent.com/AdguardTeam/AdGuardHome/master/scripts/install.sh | sh -s --
-v
sudo ./AdGuardHome -s install
sudo ./AdGuardHome -s start
```

FritzBox DNS-Settings

Zwei Einstellungen:

- Heimnetz -> Netzwerk -> Netzwerkeinstellungen -> IP-Adressen -> IPv4-Konfiguration -> Heimnetz -- für die DHCP-Verteilung
- Internet -> Zugangsdaten -> DNS-Server -- für die Angabe welchen DNS-Server die Fritzbox selbst verwendet

Blocklists

AdGuard spiegelt die Listen im eigenen GitHub Repo, das ist unschön, also in allen Listen diese Stelle anschauen und Listen-URLs entsprechend anpassen:

```
! Source: https://adguardteam.github.io/AdGuardSDNSFilter/Filters/filter.txt
```

- Standardliste von AdGuard

- AdAway (kommt von Android, primär also mobile ads)
- Dan Pollock (Security)
- Phishing
- Malware
- Badware
- EasyList Germany
- Neu registrierte Domains (<10 Tage, Vorsicht groß!)
- Drogen
- Pornowerbung
- ganze Pornoseiten

Whitelists

- Hagezi Referral (wenn man viel Internet-Shopping macht)
- uBlock unbreak (von den uBlock Origin Machern genutzt um Seiten zu reparieren, die mit der Extension sonst kaputt gehen würden)

Quellen für Listen

- Hagezi (Github)
- Dandelion Sprout (Github)
- easylist.to

Viele Listen sind klassische Adblock Listen, die ziehen bei DNS aus unten genannten Gründen nicht, weil sie keine Domains, sondern Seiteninhalte definieren. Daher muss spezifisch nach DNS-Blocklisten gesucht werden.
Adblock-Syntax wird auch interpretiert, allerdings nur die explizit erwähnten ganzen Domains.

Github Listen über CDN ziehen

Tut den Listenerstellern ein Gefallen und zieht Listen von Github über ein Gratis CDN für Github-Projekte

https://cdn.jsdelivr.net/gh/USERNAME/REPO@latest/PATH

z.B. https://blocklistproject.github.io/Lists/alt-version/drugs-nl.txt ->
https://cdn.jsdelivr.net/gh/blocklistproject/Lists@latest/alt-version/drugs-nl.txt

Upstream DNS

Erfahrungsgemäß ist die "Parallel Requests" Option etwas schneller. Ich empfehle eine größere Liste in den ersten Monaten mit Parallel Requests laufen zu lassen bis sich die für euch auf Dauer fünf schnellsten DNS herauskristallisiert haben. Dann Liste entsprechend kürzen. 30-40 Requests von denen letztendlich nur einer genommen wird ist für euch eventuell gut, erzeugt aber recht viel unnötigen overhead.

- https://dns10.quad9.net/dns-query
- tls://unfiltered.adguard-dns.com
- https://dnsforge.de/dns-query
- quic://dns.adguard.com
- quic://zero.dns0.eu
- tls://one.one.one.one
- tls://dns.google
- https://dns.google/dns-query
- tls://dot.ffmuc.net
- tls://dns.njal.la
- tls://dns.mullvad.net
- tls://recursor01.dns.lightningwirelabs.com
- https://sky.rethinkdns.com/
- tls://max.rethinkdns.com
- tls://dns.digitale-gesellschaft.ch
- https://dns.digitale-gesellschaft.ch/dns-query
- tls://dns.switch.ch
- https://dns.switch.ch/dns-query
- https://doh.applied-privacy.net/query
- tls://dot1.applied-privacy.net
- quic://dnsforge.de:853
- tls://dnsforge.de
- https://dns.nextdns.io

Average Upstream Response Time

aus 24h Statistiken

- https://dns10.quad9.net:443/dns-query - 19 ms

- tls://one.one.one.one:853 - 21 ms
- tls://dns.google:853 - 23 ms
- https://dns.google:443/dns-query - 24 ms
- tls://unfiltered.adguard-dns.com:853 - 26 ms
- tls://max.rethinkdns.com:853 - 30 ms

alles darüber ist mir zu langsam für DNS

Einschränkungen

Keine kosmetischen Filter

Einzelne Seitenelemente von Websites, z.B. Cookie-Banner, die von der gleichen Domain wie die Website auf der sie angezeigt werden sollen kommen, können auf DNS-Ebene nicht blockiert werden. Hier müsste man mit einem Zusammenspiel z.B. mit uBlock Origin oder "I don't care about cookies" arbeiten, die dann auch einzelne #div-Elemente blockieren können. Ginge auch mit der Windows-Version von Adguard, wenns Systemweit sein soll, dann muss man aber auf Feature-Dopplungen mit AdGuard Home achten.

Es gibt deshalb schon einen Proxy-Server von AdGuard, dessen Funktion aber nicht in AdGuard Home übernommen ist.

Keine Adblock-DNS als Upstream nutzen

Das Blocking passiert nun lokal bei euch, wenn ihr also als Upstream-DNS nur öffentliche Adblock-DNS nutzt kann AdGuard den lokalen Cache nicht richtig aufbauen, weil manches angefragte wegen Adblocking vom Upstream-DNS nicht beantwortet wird.

unbreak (or intentionally break) stuff

Dinge die entblockt werden mussten, oder Dinge bei denen nachgeholfen werden musste um sie auch wirklich zu blocken.

```
## COPILOT
@@||*.data.microsoft.com^$important
# there's more office things running through here. Be careful, you probably need more
subdomains to add here

## SONOS
```

```
# src: https://en.community.sonos.com/components-and-architectural-228999/sonos-no-longer-works-if-i-block-their-metric-tracking-dns-6850485
@@||msmetrics.ws.sonos.com^$important

# TUNEIN
||*.chunks.tunein.com^$important

## SHOPS
@@||*.banggood.com^$important

## LOGITECH
# src https://de.hub.sync.logitech.com/syncguides/post/firewall-and-proxy-setup-information-for-sync-Y5VKr0FBrwgr4Bg
@@||datapipeline.logitech.io^$important
@@||ghub.logitech.io^$important
@@||*.prod.logitech.io^$important
@@||lb-ghub-3024720.us-west-1.elb.amazonaws.com^$important
@@||cognito-idp.us-west-2.amazonaws.com^$important
@@||*.vc.logitech.com^$important
@@||*.sync.logitech.com^$important
@@||22ulqg35c4-dsn.algolia.net^$important

## SWAPFIETS
@@||hubspotlinks.com^$important
```

Raspberry Pi LEDs abschalten

4B

https://smarthomescene.com/guides/how-to-disable-leds-on-raspberry-pi-3b-4b/

```
#!/bin/bash
sudo cp /boot/firmware/config.txt /boot/firmware/config.txt.bak
sudo sed -i '1i\
# Turn off LEDs
dtparam=pwr_led_trigger=default-on\
dtparam=pwr_led_activelow=off\
dtparam=act_led_trigger=none\
dtparam=act_led_activelow=off\
dtparam=eth_led0=4\
dtparam=eth_led1=4\
' /boot/firmware/config.txt
```

Internet-Monitoring auf RPI

Prüft auf

- Router Verfügbarkeit
- Third Hop beim Aufrufen von Google DNS (Verfügbarkeit des Services vom Provider)
- Verfügbarkeit von DNS-Auflösung
- Möglichkeit Daten per HTTP zu bekommen

Achtet auf Installationen von PiHole oder Adguard Home, die die Prüfmethode für den Provider-Hop nach traceroute brechen würden...

Log Rotation bei 10 MB.

Raspbian bringt traceroute nicht von Haus aus mit

```
sudo apt install traceroute
```

Generator Claude Sonnet 4 für Linux 6.6.28+rpt-rpi-v8 aarch64

```
#!/bin/bash

# Internet Connection Monitor Script für Raspbian
# Überwacht Gateway-Erreichbarkeit und Internet-Konnektivität
# Autor: Generated Script
# Version: 1.0

# Konfiguration
LOG_FILE="/var/log/internet_monitor.log"
MAX_LOG_SIZE=10485760  # 10MB in Bytes
TIMEOUT=5              # Ping Timeout in Sekunden
PING_COUNT=3           # Anzahl Ping-Pakete

# DNS Server für Konnektivitätstests
DNS_SERVERS=("8.8.8.8" "1.1.1.1" "9.9.9.9")

# Farben für Terminal-Ausgabe (falls direkt ausgeführt)
RED='\033[0;31m'
GREEN='\033[0;32m'
```

```
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

# Funktion: Timestamp generieren
get_timestamp() {
    date '+%Y-%m-%d %H:%M:%S'
}

# Funktion: Log-Datei rotieren wenn zu groß
rotate_log() {
    if [[ -f "$LOG_FILE" && $(stat -f%z "$LOG_FILE" 2>/dev/null || stat -c%s "$LOG_FILE"
2>/dev/null) -gt $MAX_LOG_SIZE ]]; then
        mv "$LOG_FILE" "${LOG_FILE}.old"
        echo "$(get_timestamp) - Log-Datei rotiert" >> "$LOG_FILE"
    fi
}

# Funktion: Gateway ermitteln
get_default_gateway() {
    ip route show default | awk '/default/ { print $3; exit }'
}

# Funktion: Provider Next Hop ermitteln (echter Provider-Hop, nicht AdGuard)
get_provider_nexthop() {
    local gateway=$(get_default_gateway)
    if [[ -n "$gateway" ]]; then
        # Traceroute zum ersten externen DNS-Server
        local traceroute_output=$(traceroute -n -m 5 -w 2 8.8.8.8 2>/dev/null)

        # Durchsuche Hops 2-4 nach dem ersten nicht-privaten IP (Provider-Hop)
        for hop in {2..4}; do
            local hop_ip=$(echo "$traceroute_output" | awk -v hop="$hop" 'NR==hop+1 {print
$2}' | head -1)

            # Prüfe ob IP gesetzt und nicht leer/privat/loopback ist
            if [[ -n "$hop_ip" && "$hop_ip" != "*" && "$hop_ip" != "* * *" ]]; then
                # Prüfe ob es eine öffentliche IP ist (nicht RFC1918/Loopback/AdGuard)
                if ! is_private_ip "$hop_ip"; then
                    echo "$hop_ip"
```

```
                    return 0
                fi
            fi
        done
    fi
    echo ""
}

# Funktion: Prüfen ob IP privat/loopback/AdGuard ist
is_private_ip() {
    local ip="$1"

    # Prüfe auf private/reservierte IP-Bereiche
    if [[ "$ip" =~ ^10\. ]] || \
       [[ "$ip" =~ ^172\.(1[6-9]|2[0-9]|3[01])\. ]] || \
       [[ "$ip" =~ ^192\.168\. ]] || \
       [[ "$ip" =~ ^127\. ]] || \
       [[ "$ip" =~ ^192\.0\.0\. ]] || \
       [[ "$ip" =~ ^169\.254\. ]]; then
        return 0  # ist privat
    else
        return 1  # ist öffentlich
    fi
}

# Funktion: Netzwerk-Interface des Standard-Gateways ermitteln
get_gateway_interface() {
    ip route show default | awk '/default/ { print $5; exit }'
}

# Funktion: Ping-Test durchführen
ping_test() {
    local target="$1"
    local description="$2"

    if ping -c $PING_COUNT -W $TIMEOUT "$target" >/dev/null 2>&1; then
        echo "$(get_timestamp) - ✓ $description ($target) - ERREICHBAR" >> "$LOG_FILE"
        return 0
    else
```

```
        echo "$(get_timestamp) - ✗ $description ($target) - NICHT ERREICHBAR" >> "$LOG_FILE"
        return 1
    fi
}

# Funktion: DNS-Auflösung testen
dns_test() {
    local test_domain="google.com"

    if nslookup "$test_domain" >/dev/null 2>&1; then
        echo "$(get_timestamp) - ✓ DNS-Auflösung ($test_domain) - FUNKTIONIERT" >> "$LOG_FILE"
        return 0
    else
        echo "$(get_timestamp) - ✗ DNS-Auflösung ($test_domain) - FEHLGESCHLAGEN" >>
"$LOG_FILE"
        return 1
    fi
}

# Funktion: HTTP/HTTPS Konnektivität testen
http_test() {
    local test_url="http://www.google.com"

    if curl -s --connect-timeout $TIMEOUT --max-time $((TIMEOUT*2)) "$test_url" >/dev/null
2>&1; then
        echo "$(get_timestamp) - ✓ HTTP-Konnektivität ($test_url) - FUNKTIONIERT" >>
"$LOG_FILE"
        return 0
    else
        echo "$(get_timestamp) - ✗ HTTP-Konnektivität ($test_url) - FEHLGESCHLAGEN" >>
"$LOG_FILE"
        return 1
    fi
}

# Funktion: Netzwerk-Informationen sammeln
collect_network_info() {
    local gateway=$(get_default_gateway)
    local provider_hop=$(get_provider_nexthop)
```

```
    local interface=$(get_gateway_interface)
    local ip_addr=$(ip addr show "$interface" 2>/dev/null | grep "inet " | head -n1 | awk
'{print $2}' | cut -d'/' -f1)

    echo "$(get_timestamp) - Netzwerk-Info: Interface=$interface, IP=$ip_addr,
Gateway=$gateway, Provider-NextHop=$provider_hop" >> "$LOG_FILE"
}

# Funktion: Vollständige Verbindungsprüfung
check_connectivity() {
    echo "$(get_timestamp) - === Internet-Konnektivitätsprüfung gestartet ===" >> "$LOG_FILE"

    # Log rotieren falls nötig
    rotate_log

    # Netzwerk-Informationen sammeln
    collect_network_info

    local gateway=$(get_default_gateway)
    local provider_nexthop=$(get_provider_nexthop)
    local gateway_reachable=false
    local provider_reachable=false
    local internet_reachable=false
    local dns_working=false
    local http_working=false

    # Gateway-Erreichbarkeit prüfen
    if [[ -n "$gateway" ]]; then
        if ping_test "$gateway" "Gateway"; then
            gateway_reachable=true
        fi
    else
        echo "$(get_timestamp) - ✗ Kein Standard-Gateway gefunden!" >> "$LOG_FILE"
    fi

    # Provider Next Hop prüfen (wichtigster Test für Providerseite!)
    if $gateway_reachable; then
        if [[ -n "$provider_nexthop" ]]; then
            if ping_test "$provider_nexthop" "Provider Next Hop"; then
```

```
                provider_reachable=true
            fi
        else
            echo "$(get_timestamp) - ! Provider Next Hop konnte nicht ermittelt werden
(AdGuard/Firewall?)" >> "$LOG_FILE"
            # Fallback: Teste externe DNS direkt
            for dns_server in "${DNS_SERVERS[@]}"; do
                if ping_test "$dns_server" "Fallback Provider Test"; then
                    provider_reachable=true
                    break
                fi
            done
        fi
    fi

    # DNS-Server-Erreichbarkeit prüfen (falls Provider erreichbar)
    if $provider_reachable || ($gateway_reachable && [[ -z "$provider_nexthop" ]]); then
        for dns_server in "${DNS_SERVERS[@]}"; do
            if ping_test "$dns_server" "DNS-Server"; then
                internet_reachable=true
                break
            fi
        done
    fi

    # DNS-Auflösung testen
    if $internet_reachable; then
        if dns_test; then
            dns_working=true
        fi
    fi

    # HTTP-Konnektivität testen
    if $dns_working; then
        if http_test; then
            http_working=true
        fi
    fi
```

```
    # Gesamtstatus bestimmen
    local status="UNBEKANNT"
    local status_code=3

    if $http_working; then
        status="VOLLSTÄNDIG FUNKTIONSFÄHIG"
        status_code=0
    elif $dns_working; then
        status="DNS FUNKTIONIERT, HTTP PROBLEME"
        status_code=1
    elif $internet_reachable; then
        status="INTERNET ERREICHBAR, DNS PROBLEME"
        status_code=1
    elif $provider_reachable; then
        status="PROVIDER ERREICHBAR, INTERNET PROBLEME"
        status_code=2
    elif $gateway_reachable; then
        status="NUR LOKALES NETZWERK - PROVIDER NICHT ERREICHBAR"
        status_code=2
    else
        status="KEINE NETZWERKVERBINDUNG"
        status_code=3
    fi

    echo "$(get_timestamp) - STATUS: $status (Code: $status_code)" >> "$LOG_FILE"
    echo "$(get_timestamp) - === Prüfung beendet ===" >> "$LOG_FILE"
    echo "" >> "$LOG_FILE"

    # Status auch auf stdout ausgeben (für manuelle Ausführung)
    if [[ -t 1 ]]; then
        case $status_code in
            0) echo -e "${GREEN}✓ Internet: $status${NC}" ;;
            1) echo -e "${YELLOW}⚠ Internet: $status${NC}" ;;
            2) echo -e "${YELLOW}⚠ Internet: $status${NC}" ;;
            3) echo -e "${RED}✗ Internet: $status${NC}" ;;
        esac
    fi

    return $status_code
```

```
}

# Funktion: Hilfe anzeigen
show_help() {
    echo "Internet Connection Monitor Script"
    echo ""
    echo "Verwendung: $0 [OPTION]"
    echo ""
    echo "Optionen:"
    echo "  -h, --help     Zeige diese Hilfe"
    echo "  -l, --log      Zeige die letzten Log-Einträge"
    echo "  -s, --status   Führe eine einmalige Statusprüfung durch"
    echo "  -c, --config   Zeige aktuelle Konfiguration"
    echo ""
    echo "Standard-Verhalten: Führe Konnektivitätsprüfung durch"
    echo ""
    echo "Log-Datei: $LOG_FILE"
}

# Funktion: Log-Einträge anzeigen
show_log() {
    local lines=${1:-50}
    if [[ -f "$LOG_FILE" ]]; then
        echo "Letzte $lines Zeilen aus $LOG_FILE:"
        echo "=" $(printf '=%.0s' {1..60})
        tail -n "$lines" "$LOG_FILE"
    else
        echo "Log-Datei $LOG_FILE nicht gefunden."
    fi
}

# Funktion: Konfiguration anzeigen
show_config() {
    echo "Aktuelle Konfiguration:"
    echo "======================="
    echo "Log-Datei: $LOG_FILE"
    echo "Max. Log-Größe: $((MAX_LOG_SIZE/1024/1024)) MB"
    echo "Ping-Timeout: $TIMEOUT Sekunden"
    echo "Ping-Count: $PING_COUNT Pakete"
```

```
    echo "DNS-Server: ${DNS_SERVERS[*]}"
    echo ""
    echo "Aktuelles Gateway: $(get_default_gateway)"
    echo "Gateway-Interface: $(get_gateway_interface)"
    echo "Provider Next Hop: $(get_provider_nexthop)"
}

# Hauptprogramm
main() {
    # Root-Rechte prüfen (für Log-Datei in /var/log)
    if [[ $EUID -ne 0 && "$LOG_FILE" == /var/log/* ]]; then
        echo "Warnung: Root-Rechte benötigt für Log-Datei in /var/log/"
        echo "Verwende temporäre Log-Datei: /tmp/internet_monitor.log"
        LOG_FILE="/tmp/internet_monitor.log"
    fi

    # Parameter verarbeiten
    case "${1:-}" in
        -h|--help)
            show_help
            exit 0
            ;;
        -l|--log)
            show_log "${2:-50}"
            exit 0
            ;;
        -s|--status)
            check_connectivity
            exit $?
            ;;
        -c|--config)
            show_config
            exit 0
            ;;
        "")
            check_connectivity
            exit $?
            ;;
        *)
```

```
            echo "Unbekannte Option: $1"
            echo "Verwende '$0 --help' für Hilfe."
            exit 1
            ;;
    esac
}

# Script ausführen
main "$@"
```

Installation

```
sudo cp internet_monitor.sh /usr/local/bin/
sudo chmod +x /usr/local/bin/internet_monitor.sh
```

Test

```
sudo /usr/local/bin/internet_monitor.sh --status
```

Cronjob

```
*/5 * * * * /usr/local/bin/internet_monitor.sh >/dev/null 2>&1
```