NGINX Proxy Manager – IPv64-IP-Blacklist

Stand: 07.11.2026 **System:** Proxmox-LXC `nginx`, darin NGINX Proxy Manager über Docker Compose **Ziel:** Bekannte Angreifer-IP-Adressen und -Netzwerke aus der IPv64-/blocklist.de-Liste für alle NPM-Proxy-Hosts mit HTTP 403 abweisen. Automatische Aktualisierung alle vier Stunden einschließlich Prüfung, Sicherung und Protokollrotation.

1. Ergebnis / Status

Komponente	Stand nach Einrichtung
NGINX Proxy Manager	Docker-Container `nginx-app-1` (OpenResty/NGINX)
Blacklist	Aktiv; 26.381 Einträge beim erfolgreichen Test am 06.11.2026 (Anzahl ändert sich)
Sperrmechanismus	NGINX `geo` + `if ($ipv64_blocked) { return 403; }`
Gültigkeitsbereich	Alle regulären NPM-Proxy-Hosts; nicht automatisch NPM-Adminoberfläche, Redirection Hosts oder TCP/UDP-Streams
Update	Cron täglich um Minute 17 alle vier Stunden (Zeitzone des LXC beachten)
Updateschutz	HTTPS, IP-Validierung, Mindestanzahl, Vergleich, Konfigurationstest, Sicherung und Rollback
Logging	`/var/log/ipv64-nginx-update.log`, wöchentlich rotiert
Funktionstest	Erlaubte Test-IP → HTTP 302, gesperrte Test-IP → HTTP 403

Hinweis: Die Uhrzeit des Update-Skripts erschien in UTC. Die genauen lokalen Cron-Ausführungszeiten richten sich nach der Zeitzone des LXC (prüfen mit `timedatectl`).

2. Voraussetzungen / Docker-Umgebung

Auf dem LXC-Host als `root` ausführen, **nicht** auf dem Proxmox-Host:

```
docker ps --format 'table {{.Names}}\t{{.Image}}\t{{.Status}}'
docker inspect nginx-app-1 --format '{{range .Mounts}}{{println .Source "->" .Destination}}{{end}}'
docker exec nginx-app-1 nginx -t
```

Ermittelte Container und Mounts:

```
nginx-app-1   jc21/nginx-proxy-manager:latest
nginx-db-1    jc21/mariadb-aria:latest
duckdns       lscr.io/linuxserver/duckdns:latest

/root/data        -> /data
/root/letsencrypt -> /etc/letsencrypt
```

Wichtig: Die IP-Blacklist wird in den persistenten NPM-Daten unter `/root/data` gespeichert. Die Docker-Compose-Datei und die automatisch erzeugten Proxy-Host-Konfigurationen werden nicht

verändert.

3. CA-Zertifikatsfehler beim Download (einmalige Reparatur)

Symptom: Der ursprüngliche Download scheiterte mit:

```
curl: (77) error setting certificate file: /etc/ssl/certs/ca-certificates.crt
```

Die CA-Datei war vorhanden; das Paket `ca-certificates` war installiert. Ein erster Neuaufbau mit `update-ca-certificates –fresh` beseitigte den Fehler noch nicht.

Die Diagnose zeigte:

- Mit einem einzelnen vertrauenswürdigen CA-Zertifikat funktionierte HTTPS:

```
curl -q -v -I --cacert /usr/share/ca-certificates/mozilla/ISRG_Root_X1.crt https://ipv64.net/
```

- Die Prüfung des gesamten Bundles scheiterte mit ASN.1-/PEM-Fehlern:

```
openssl verify -CAfile /etc/ssl/certs/ca-certificates.crt /usr/share/ca-certificates/mozilla/ISRG_Root_X1.crt
```

- Unter `/usr/local/share/ca-certificates/` lag eine fehlerhafte Datei `digicert_global_root_g2.crt` (Kleinschreibung). Eine ähnlich benannte Datei mit Großschreibung war lesbar.

Durchgeführte Reparatur: Nur die konkret fehlerhafte Datei wurde gesichert und aus der lokalen CA-Importstruktur entfernt; es wurde nichts gelöscht.

```
mkdir -p /root/ca-quarantaene
mv /usr/local/share/ca-certificates/digicert_global_root_g2.crt \
  /root/ca-quarantaene/
update-ca-certificates --fresh
openssl verify -CAfile /etc/ssl/certs/ca-certificates.crt \
  /usr/share/ca-certificates/mozilla/ISRG_Root_X1.crt
curl -q -I --max-time 20 https://ipv64.net/
```

Ergebnis: `ISRG_Root_X1.crt: OK` und `HTTP/1.1 200 OK`.

Die weiteren Meldungen zu `GeoTrust_TLS_RSA_CA_G1`, `strato_bundle` und doppelten Zertifikaten waren Warnungen, nicht die Ursache des hier behobenen Fehlers. Diese Dateien nicht pauschal löschen. `curl -k` wurde ausdrücklich nicht verwendet, damit die TLS-Prüfung erhalten bleibt.

4. IP-Quelle und erste Aufbereitung

Quelle: IPv64 blocklist.de – all

Die Quelle enthält IP-Adressen und CIDR-Netze (IPv4/IPv6), **keine DNS-Domain-Blockliste**. Deshalb ist sie für NGINX geeignet, nicht als gewöhnliche DNS-Blockliste in AdGuard Home. Einen entsprechenden AdGuard-Eintrag wieder entfernen.

Erster Download und Umwandlung in NGINX-geo-Syntax:

```
mkdir -p /root/data/nginx/custom
curl -fLsS --retry 3 --max-time 90 \
  https://ipv64.net/blocklists/ipv64_blocklist_blocklistde_all.txt \
  -o /root/data/nginx/ipv64-source.tmp

awk 'NF == 1 && /^[0-9A-Fa-f:.]+(\/[0-9]+)?$/ {print $1 " 1;"}' \
  /root/data/nginx/ipv64-source.tmp \
  > /root/data/nginx/ipv64-geo.conf.tmp

wc -l /root/data/nginx/ipv64-source.tmp
wc -l /root/data/nginx/ipv64-geo.conf.tmp
head -n 5 /root/data/nginx/ipv64-geo.conf.tmp

install -m 644 /root/data/nginx/ipv64-geo.conf.tmp \
  /root/data/nginx/ipv64-geo.conf
```

Beim ersten Import: 26.375 Einträge; beim folgenden Updater-Lauf: 26.381. Die späteren automatischen Updates validieren die IP-/Netzsyntax mit Pythons `ipaddress`-Modul (deutlich zuverlässiger als nur der erste AWK-Filter).

Beispiel eines geo-Eintrags:

```
1.14.192.95/32 1;
```

5. NGINX Proxy Manager – globale Konfiguration

NPM erlaubt die persistenten Custom-Includes `/data/nginx/custom/http_top.conf` (im http-Kontext) und `/data/nginx/custom/server_proxy.conf` (in den Proxy-Host-Serverblöcken).

5.1. GEO-Variable im HTTP-Kontext definieren

Host-Datei: `/root/data/nginx/custom/http_top.conf`

```
geo $ipv64_blocked {
    default 0;
    include /data/nginx/ipv64-geo.conf;
}
```

5.2. Sperrregel für alle regulären Proxy Hosts

Host-Datei: `/root/data/nginx/custom/server_proxy.conf`

```
# IPv64 Blacklist – gesperrte Clients bekommen HTTP 403
if ($ipv64_blocked) {
    return 403;
}
```

5.3. Prüfen und aktivieren

```
docker exec nginx-app-1 nginx -t
docker exec nginx-app-1 nginx -s reload

docker exec nginx-app-1 nginx -T 2>/dev/null | \
  grep -F -e 'geo $ipv64_blocked' \
          -e 'include /data/nginx/ipv64-geo.conf;' \
          -e 'if ($ipv64_blocked)' | sort -u
```

Ergebnis: Konfiguration gültig und Reload erfolgreich. Die Warnungen `listen … http2 directive is deprecated` in `proxy_host/6.conf` sind unabhängig von der Blacklist und verhinderten den Reload nicht.

6. Automatisches Update mit Prüfung und Rollback

Installiertes Skript: `/usr/local/sbin/update-ipv64-nginx.sh`

```
#!/bin/bash
set -Eeuo pipefail

DIR="/root/data/nginx"
TARGET="$DIR/ipv64-geo.conf"
BACKUP="$DIR/ipv64-geo.conf.bak"
URL="https://ipv64.net/blocklists/ipv64_blocklist_blocklistde_all.txt"
CONTAINER="nginx-app-1"

exec 9>/run/lock/ipv64-nginx.lock
flock -n 9 || {
    echo "Update läuft bereits."
    exit 0
}

RAW=$(mktemp "$DIR/.ipv64-raw.XXXXXX")
NEW=$(mktemp "$DIR/.ipv64-new.XXXXXX")
RESTORE=$(mktemp "$DIR/.ipv64-restore.XXXXXX")
trap 'rm -f "$RAW" "$NEW" "$RESTORE"' EXIT

echo "=== IPv64 Update: $(date) ==="
```

```
curl -fLsS --retry 3 --max-time 90 \
    "$URL" -o "$RAW"

python3 - "$RAW" "$NEW" "$TARGET" <<'PY'
import ipaddress
import sys

source, output, previous = sys.argv[1:]
entries = []
seen = set()

with open(source, encoding="utf-8-sig") as f:
    for line in f:
        line = line.strip()
        if not line or line.startswith("#"):
            continue

        network = ipaddress.ip_network(line, strict=False)

        if network.prefixlen == 0:
            raise ValueError("Unsicheres /0-Netz erkannt")

        entry = str(network)
        if entry not in seen:
            seen.add(entry)
            entries.append(entry)

with open(previous, encoding="utf-8") as f:
    old_count = sum(
        1 for line in f if line.strip()
    )

if len(entries) < 10000:
    raise ValueError("Zu wenige Einträge")

if len(entries) < old_count // 2:
    raise ValueError("Ungewöhnlich starker Listenrückgang")

with open(output, "w", encoding="utf-8") as f:
    for entry in entries:
        f.write(f"{entry} 1;\n")

print(f"Alte Einträge: {old_count}")
print(f"Neue Einträge: {len(entries)}")
PY

if cmp -s "$TARGET" "$NEW"; then
    echo "Keine Änderungen erforderlich."
    exit 0
fi
```

```
cp -p "$TARGET" "$BACKUP"
chmod 644 "$NEW"
mv -f "$NEW" "$TARGET"

if docker exec "$CONTAINER" nginx -t -q &&
   docker exec "$CONTAINER" nginx -s reload; then
    echo "ERFOLG: Blacklist aktualisiert."
else
    echo "FEHLER: Vorherige Liste wird wiederhergestellt."
    cp -p "$BACKUP" "$RESTORE"
    mv -f "$RESTORE" "$TARGET"
    docker exec "$CONTAINER" nginx -t -q &&
        docker exec "$CONTAINER" nginx -s reload
    exit 1
fi
```

Installation und manueller Test:

```
chmod 750 /usr/local/sbin/update-ipv64-nginx.sh
bash -n /usr/local/sbin/update-ipv64-nginx.sh
/usr/local/sbin/update-ipv64-nginx.sh
```

Erfolgreicher Test: 26.375 → 26.381 Einträge, NGINX-Konfiguration geprüft, Reload erfolgreich. Ein zweiter Lauf mit unveränderten 26.381 Einträgen meldete `Keine Änderungen erforderlich.`

Hinweis zur Fehlerbehandlung: Das Skript bewahrt eine Sicherung der letzten Version unter `/root/data/nginx/ipv64-geo.conf.bak` auf und versucht bei Fehlern nach dem Austausch die vorherige Liste wiederherzustellen. Ein provozierter Fehler-/Rollback-Test wurde nicht durchgeführt. Bei einem vollständigen Ausfall des Docker-Dienstes kann auch der Wiederherstellungs-Reload fehlschlagen; dann ist manuelles Eingreifen erforderlich.

7. Cron: Aktualisierung alle vier Stunden

Datei `/etc/cron.d/ipv64-nginx`:

```
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

17 */4 * * * root /usr/local/sbin/update-ipv64-nginx.sh >> /var/log/ipv64-nginx-update.log 2>&1
```

Aktivieren / prüfen:

```
chmod 644 /etc/cron.d/ipv64-nginx
systemctl enable --now cron
systemctl is-active cron
cat /etc/cron.d/ipv64-nginx
```

```
timedatectl
```

`cron` meldete `active`. Die Ausführung findet um `00:17, 04:17, 08:17, 12:17, 16:17, 20:17` **in der eingestellten Zeitzone des LXC** statt. Bei UTC-Zeitzone entsprechen die deutschen Lokalzeiten je nach Sommer-/Winterzeit anderen Uhrzeiten.

8. Logrotate

Datei `/etc/logrotate.d/ipv64-nginx`:

```
/var/log/ipv64-nginx-update.log {
    su root root
    weekly
    rotate 8
    compress
    missingok
    notifempty
    create 0640 root adm
}
```

Der erste Test schlug fehl, weil `/var/log` die Berechtigung `775 root:syslog` besitzt. Das ist nicht automatisch ein Problem; **die Rechte von /var/log wurden nicht geändert**. Die Ergänzung `su root root` behob die Logrotate-Fehlermeldung.

Prüfung:

```
logrotate -d /etc/logrotate.d/ipv64-nginx 2>&1 | tail -n 20
/usr/local/sbin/update-ipv64-nginx.sh >> /var/log/ipv64-nginx-update.log
2>&1
tail -n 15 /var/log/ipv64-nginx-update.log
logrotate -d /etc/logrotate.d/ipv64-nginx 2>&1 | tail -n 20
```

Ergebnis: Logdatei erstellt; die Debug-Prüfung (`-d`, ohne wirkliche Rotation) war anschließend fehlerfrei.

9. Funktionstest: HTTP 403

Test von der NGINX-LXC-Konsole mit einer tatsächlich bei NPM eingerichteten HTTPS-Domain. Die Domain muss korrekt geschrieben sein und ein gültiges Zertifikat besitzen. `–resolve` verbindet lokal über Port 443, erhält aber den für SNI notwendigen Hostnamen.

```
DOMAIN="joplin.DEINE-DOMAIN.de"  # Platzhalter ersetzen

BLOCKED_IP=$(awk '
  $1 ~ /^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+\/32$/ {
    sub(/\/32$/, "", $1)
    print $1
```

```
    exit
  }
' /root/data/nginx/ipv64-geo.conf)

echo "Gesperrte Test-IP: $BLOCKED_IP"

curl --noproxy '*' -sS --max-time 15 \
  --resolve "$DOMAIN:443:127.0.0.1" \
  -H 'X-Real-IP: 203.0.113.10' \
  -o /dev/null -w 'HTTP: %{http_code}\n' "https://$DOMAIN/"

curl --noproxy '*' -sS --max-time 15 \
  --resolve "$DOMAIN:443:127.0.0.1" \
  -H "X-Real-IP: $BLOCKED_IP" \
  -o /dev/null -w 'HTTP: %{http_code}\n' "https://$DOMAIN/"
```

Beobachtetes Ergebnis: Erlaubte IP → `302` (normale Joplin-Weiterleitung); gesperrte IP `1.14.192.95` → `403`. Die Prüfung der echten Sperrfunktion war damit erfolgreich.

Zwischenfall beim ersten Test: Mit einem nicht passenden Hostnamen meldete `curl` `tlsv1 unrecognized name` / `HTTP 000` (SNI-Problem). Nach Verwendung der tatsächlichen Joplin-Domain funktionierte der Test.

10. Test der Real-IP-Verarbeitung von außen

NPM enthält automatisch konfigurierte `real_ip_header X-Real-IP`- und `set_real_ip_from`-Einträge für private Netze sowie Cloudflare-/CloudFront-Netze. Diese IP-Bereiche sind **vertrauenswürdige Proxys, keine Blacklist**.

Die Logs zeigten lokale Clients (z. B. `192.168.10.x`) und externe öffentliche Client-IP-Adressen. Das ist ein gutes Indiz für eine funktionierende IP-Erfassung, aber keine vollständige Sicherheitsgarantie.

Test **vom MacBook über Mobilfunk-Hotspot**, nicht aus dem LAN:

```
DOMAIN="joplin.DEINE-DOMAIN.de"  # Platzhalter ersetzen

curl -sS -o /dev/null -w 'HTTP: %{http_code}\n' "https://$DOMAIN/"
curl -sS -o /dev/null \
  -H 'X-Real-IP: 1.14.192.95' \
  -w 'HTTP: %{http_code}\n' "https://$DOMAIN/"
```

Beobachtet: `302` ohne Header und `302` mit gefälschtem Header. Die Anfrage wurde über diesen externen Weg also nicht fälschlich als die blockierte IP eingestuft.

Grenze des Tests: Gleiche Statuscodes beweisen noch nicht, dass sämtliche Proxy-/Header-Konstellationen gegen Spoofing abgesichert sind. Werden später Cloudflare, ein weiterer Reverse Proxy oder besondere Portweiterleitungen eingesetzt, `real_ip` erneut prüfen.

11. Wartung und Kontrolle

```
# Anzahl gesperrter IPs/Netze
wc -l /root/data/nginx/ipv64-geo.conf

# Letzte Updater-Meldungen
tail -n 50 /var/log/ipv64-nginx-update.log

# Manueller Update-Lauf
/usr/local/sbin/update-ipv64-nginx.sh

# Cron und NPM prüfen
systemctl is-active cron
docker ps --filter name=nginx-app-1
docker exec nginx-app-1 nginx -t

# Letzte Zugriffe (ggf. öffentliche IPs vor Weitergabe anonymisieren)
tail -n 30 /root/data/logs/proxy-host-1_access.log
```

Zur Erinnerung: Die Blockliste schützt vor bekannten Quell-IPs, ersetzt aber **keine** Authentifizierung, Aktualisierung der Dienste, Zugriffsbeschränkungen für Verwaltungsoberflächen oder eine Firewall. Je nach IP-Listenqualität sind Fehlalarme möglich.

12. Sperre vorübergehend deaktivieren

Falls legitime Nutzer unerwartet HTTP 403 erhalten, lässt sich die globale Sperrregel ohne Löschen der heruntergeladenen Liste deaktivieren:

```
mv /root/data/nginx/custom/server_proxy.conf \
  /root/data/nginx/custom/server_proxy.conf.disabled

docker exec nginx-app-1 nginx -t && \
  docker exec nginx-app-1 nginx -s reload
```

Wieder aktivieren:

```
mv /root/data/nginx/custom/server_proxy.conf.disabled \
  /root/data/nginx/custom/server_proxy.conf

docker exec nginx-app-1 nginx -t && \
  docker exec nginx-app-1 nginx -s reload
```

Achtung: Der Cronjob aktualisiert die IP-Datei auch bei deaktivierter Sperrregel weiter, solange er nicht separat deaktiviert wird.

13. Wichtige Pfade

Pfad im LXC-Host	Zweck
`/root/data/nginx/ipv64-geo.conf`	Aktive IP-Liste in NGINX-geo-Syntax
`/root/data/nginx/ipv64-geo.conf.bak`	Sicherung der Liste vor dem letzten Änderungslauf
`/root/data/nginx/custom/http_top.conf`	Globale geo-Definition
`/root/data/nginx/custom/server_proxy.conf`	HTTP-403-Regel für Proxy Hosts
`/usr/local/sbin/update-ipv64-nginx.sh`	Update-, Validierungs- und Rollback-Skript
`/etc/cron.d/ipv64-nginx`	Update-Zeitplan
`/etc/logrotate.d/ipv64-nginx`	Logrotation
`/var/log/ipv64-nginx-update.log`	Update-Protokoll
`/root/ca-quarantaene/digicert_global_root_g2.crt`	Gesicherte defekte CA-Datei
`/root/data/logs/proxy-host-*_access.log`	Zugriffsprotokolle der Proxy Hosts

14. Hinweise / offene Ergänzungen

- **Sperrstatistik:** Eine Übersicht über die Anzahl tatsächlicher 403-Sperren durch diese Liste ist noch nicht eingerichtet. Ein HTTP 403 kann auch aus anderen Gründen entstehen.
- **Eigene Allow-/Denylist:** Nicht eingerichtet; bei Bedarf getrennt planen und mit klarer Priorität zu den IPv64-Einträgen einbauen.
- **Cron-Erstlauf:** Der Updater wurde manuell erfolgreich getestet. Die erste selbstständige Cron-Ausführung zum nächsten geplanten Zeitpunkt ist anhand der Logdatei zu kontrollieren.
- **Fehlerfall des Skripts:** Rollback-Logik vorhanden, aber nicht absichtlich unter Störung getestet.
- **NPM-Updates:** Dateien unter `/root/data/nginx/custom/` bleiben durch das persistente Docker-Volume erhalten; nach NPM-Upgrades trotzdem `nginx -t` und den 403-Test wiederholen.
- **HTTPS-/Strato-Zertifikate:** Die Reparatur betraf den CA-Truststore des LXC-Systems, nicht die NPM-Let's-Encrypt-Konfiguration unter `/root/letsencrypt`.

15. Kurzfazit

Die IPv64/blocklist.de-IP-Liste ist im Docker-basierten NGINX Proxy Manager erfolgreich eingebunden. Ein gesperrter Testclient erhielt HTTP 403, ein erlaubter Client die normale HTTP-302-Antwort. Der Updater läuft manuell korrekt, Cron und Logrotate sind eingerichtet. Die Konfiguration gilt global für NPM-Proxy-Hosts; andere Dienste und die NPM-Adminoberfläche sind dadurch nicht automatisch abgedeckt.

From:
https://wiki.mahlen.biz/ - **Smart-Home Wiki**

Permanent link:
https://wiki.mahlen.biz/doku.php?id=netzwerk:netzwerk_proxy:proxy_nginxproxymanager:blacklist_anlegen

Last update: **07.11.2026**