



Bau dir deinen eigenen privaten DNS-Resolver

DoH + DoT + Filter, selbst gehostet

The Morpheus

patreon.com/themorpheus

Bau dir deinen eigenen privaten DNS-Resolver (DoH + DoT + Filter)

[Was du baust](#)

[Schritt 0 — Erster SSH-Zugriff und Key-Authentifizierung](#)

[Schritt 1 — Erst die Firewall](#)

[Schritt 2 — Unbound installieren und konfigurieren](#)

[Schritt 3 — Port 53 für AdGuard Home freiräumen](#)

[Schritt 4 — AdGuard Home installieren](#)

[Schritt 5 — TLS-Zertifikat \(Let's Encrypt\)](#)

[Schritt 6 — AdGuard Home konfigurieren](#)

[Schritt 6.5 — Zugriff einschränken \(Pflicht, nicht optional\)](#)

[Schritt 6.6 — Admin-UI absichern \(auch Pflicht\)](#)

[Schritt 7 — Alles verifizieren](#)

[Client-Setup](#)

[Den Server betreiben](#)

[Sicherheits-Tipps](#)

[Was du im Hinterkopf haben solltest](#)

[Häufige Probleme und Fixes](#)

[Was du NICHT tun solltest](#)

[Kosten und Zeit](#)

[Schnellreferenz: Ports und Dateien](#)

[Die sechs Dinge, die wirklich zählen](#)

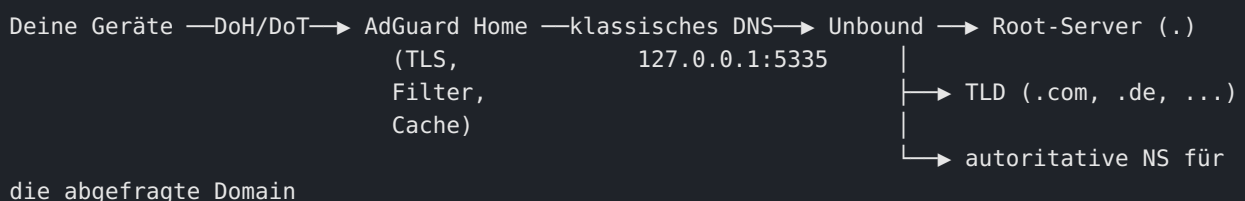
Bau dir deinen eigenen privaten DNS-Resolver (DoH + DoT + Filter)

Eine vollständige Anleitung, um einen selbst gehosteten verschlüsselten DNS-Server mit Werbe-/Tracker-Filter zu betreiben. Der gesamte Stack ist Open Source. Am Ende dieses Guides hast du einen DNS-Endpunkt, den du von deinen eigenen Geräten überall aus erreichen kannst — kein zentraler Public-Resolver wie Cloudflare oder Google sieht mehr deinen gesamten DNS-Verlauf.

Wichtig vorab — was das hier *nicht* ist. Ohne den Allowlist-Schritt aus Schritt 6.5 ist dein DoH/DoT-Endpunkt für jeden im Internet nutzbar. Das ist kein klassisches Amplification-Risiko (DoH/DoT laufen über TCP/TLS), aber Fremde könnten deinen Resolver belasten und deine IP-Reputation versauen. „Privat“ wird das Ganze erst durch die Zugriffskontrolle in Schritt 6.5. Außerdem: DNS-Verschlüsselung schützt **Client → Resolver**, sie ist keine Anonymisierung. Deine VPS-Infrastruktur sowie autoritative Nameserver sehen weiterhin Teile der Auflösungskette (siehe RFC 9076).

Was du baust

Stack: AdGuard Home (DoH/DoT-Frontend + Filter) → Unbound (rekursiver Resolver) → die echte autoritative DNS-Hierarchie.



- **AdGuard Home** ist der nach außen sichtbare Dienst. Er terminiert TLS für DoH und DoT, filtert Werbung/Tracker/Malware, cached, und leitet Misses an Unbound weiter.
- **Unbound** ist ein rekursiver Resolver, der nur auf Loopback lauscht. Er läuft die DNS-Hierarchie selbst ab, beginnend bei den Root-Servern, und validiert DNSSEC. Es gibt keinen klassischen “Upstream” — genau das ist der Privacy-Gewinn gegenüber Cloudflare/Google/Quad9 als zentralem Resolver.
- **Plain DNS auf Port 53.** AdGuard bindet den Port technisch auf allen Interfaces, aber die Firewall blockiert UDP/TCP 53 von außen. Ergebnis: kein klassischer UDP-Amplification-Vektor. Nach erfolgreichem DoH/DoT-Setup kannst du Plain DNS in AdGuard zusätzlich deaktivieren oder auf interne Interfaces beschränken.

Warum überhaupt

- **Privacy-Fan-Out (Distribution, nicht Anonymisierung).** Mit einem öffentlichen Resolver wie 1.1.1.1 sieht Cloudflare jede Domain, die du jemals abfragst. Hier verteilt sich die Sichtbarkeit: Anfragen für cloudflare.com gehen zu Cloudflares autoritativen Servern, Anfragen für bahn.de zu DENIC + den Bahn-Nameservern — jeder Betreiber sieht nur Anfragen für seine eigene Zone. Dein VPS-Anbieter und der Netzpfad sehen weiterhin den Traffic von Unbound zu autoritativen Servern (das ist klassisches DNS). QNAME-Minimization (RFC 9156) reduziert die preisgegebenen Daten, anonymisiert aber nichts.
- **Verschlüsselter Transport (Client → Resolver).** DoH und DoT bedeuten: dein ISP kann auf der Leitung weder mitlesen noch DNS-Antworten manipulieren. Das schützt nicht den Resolver→Auth-Pfad.
- **Filter.** Werbung, Tracker, Telemetrie, bekannte Malware-Domains — auf DNS-Ebene blockiert für jedes Gerät, das deinen Resolver nutzt.
- **DNSSEC-Validierung.** Unbound prüft Signaturen vom Root-Trust-Anchor abwärts. Gefälschte Antworten werden verworfen.
- **Kein UDP-Amplification-Vektor**, weil Unbound nur auf 127.0.0.1 lauscht und Port 53 von außen geblockt ist.

Was du brauchst

- Einen VPS — jeder Anbieter geht (Hostinger, Infomaniak, Hetzner, OVH, Netcup, etc.). 1 vCPU + 1 GB RAM reicht für den Privatgebrauch; 2 vCPU + 2 GB ist komfortabel. Dieser Guide geht von Ubuntu 24.04 LTS oder Debian 12 aus.
- Eine Domain, die dir gehört. Eine Subdomain wie dns.example.com reicht völlig.
- Einen A-Record, der diese Subdomain auf die öffentliche IP deines VPS zeigt. Wenn dein VPS auch IPv6 hat und du das willst, zusätzlich einen AAAA-Record. Wenn du IPv6 nicht aktiv nutzen willst, **keinen** AAAA-Record setzen — sonst hast du eventuell einen sauber gefilterten IPv4-Stack und einen offenen IPv6-Stack.
- **Cloudflare-Falle:** Wenn deine Zone bei Cloudflare liegt, muss der Record für dns.example.com auf “DNS only” / graue Wolke stehen. Mit aktivem Proxy (orange Wolke) terminiert Cloudflare TLS für deine DoH-Anfragen — d. h. Cloudflare sieht alles, und das Ziel “kein zentraler Anbieter dazwischen” ist hin. DoT auf Port 853 funktioniert über den Cloudflare-Proxy ohnehin nicht.
- SSH-Zugriff auf den VPS.
- Ungefähr 30 Minuten.

Schritt 0 — Erster SSH-Zugriff und Key-Authentifizierung

Die meisten VPS-Anbieter geben dir ein Root-Passwort (oder einen vorinstallierten SSH-Key) per Mail oder im Control-Panel. Verbinde dich erstmal:

```
ssh root@DEIN.VPS.IP
```

Frischen SSH-Key auf deinem Rechner erzeugen

Falls du noch keinen für diesen Server hast:

```
ssh-keygen -t ed25519 -f ~/.ssh/dein_vps_key -C "du@example.com"
```

Eine leere Passphrase nur, wenn du dem Speicher auf deinem lokalen Rechner vertraust. Sonst Passphrase setzen und `ssh-agent` benutzen.

Public Key auf den Server kopieren

Von deinem lokalen Rechner aus:

```
ssh-copy-id -i ~/.ssh/dein_vps_key.pub root@DEIN.VPS.IP
```

Wenn `ssh-copy-id` nicht verfügbar ist:

```
cat ~/.ssh/dein_vps_key.pub | ssh root@DEIN.VPS.IP \  
'mkdir -p ~/.ssh && chmod 700 ~/.ssh && cat >> ~/.ssh/authorized_keys && chmod 600  
~/.ssh/authorized_keys'
```

Trag dir in `~/.ssh/config` einen Eintrag ein, damit du nicht jedes Mal alle Details tippen musst:

```
Host mein-dns  
  HostName DEIN.VPS.IP  
  User root  
  IdentityFile ~/.ssh/dein_vps_key  
  IdentitiesOnly yes
```

Test:

```
ssh mein-dns 'echo OK'
```

Den privaten Key irgendwo Sicheres backuppen

Wenn du ihn verlierst und Passwort-Auth deaktiviert ist (nächster Schritt), bist du ausgesperrt und musst die Notfall-Konsole des Anbieters benutzen. Kopier `~/.ssh/dein_vps_key` in einen Passwort-Manager, auf einen verschlüsselten USB-Stick oder ein Backup-Laufwerk. Behandle ihn wie ein Passwort.

Passwort-Login + direkten Root-Passwort-Login deaktivieren

Auf dem Server:

```
cat > /etc/ssh/sshd_config.d/99-hardening.conf <<EOF
PasswordAuthentication no
PermitRootLogin prohibit-password
PubkeyAuthentication yes
KbdInteractiveAuthentication no
EOF
sshd -t && systemctl reload ssh
```

`prohibit-password` erlaubt Root-Login per Key, blockiert aber Passwort-Versuche. Verifiziere, indem du versuchst, dich mit dem Passwort einzuloggen — das sollte fehlschlagen.

Warum jetzt? Brute-Force-Scanner hauen Port 22 innerhalb von Minuten an, sobald ein frischer Server online geht. Bring den Key zum Laufen und schalte Passwort-Auth ab, bevor du irgendetwas anderes machst.

Schritt 1 — Erst die Firewall

Die Kiste dichtmachen, bevor du Dienste installierst, die auf Ports lauschen.

```
apt update && apt install -y ufw

ufw default deny incoming
ufw default allow outgoing
ufw allow 22/tcp    comment "SSH"
ufw allow 80/tcp    comment "HTTP-01 ACME (Cert-Erneuerung)"
ufw allow 443/tcp   comment "DoH"
# 853/tcp (DoT) bewusst NICHT auf – siehe Mobile-Block unten.
# Wenn du später Wildcard-Cert für DoT-Subdomain-ClientIDs nutzt:
#   ufw allow 853/tcp comment "DoT"
ufw --force enable
ufw status verbose
```

Port 53 bleibt nach außen geschlossen — dein Resolver ist für *dich*, erreichbar über DoH/DoT. Wer auf der öffentlichen IP plain DNS abfragt, läuft ins Leere.

Empfehlung: Port 853 (DoT) erstmal **nicht öffnen**. Mit ClientID-Allowlist ist DoT eh nicht direkt nutzbar (außer mit Wildcard-Cert), und für Android empfehlen wir später ohnehin VPN oder DoH-App statt nativer “Privates DNS”-Funktion. Wenn du dich später für Wildcard-Cert + DoT-Subdomain entscheidest, kannst du 853 immer noch nachträglich aufmachen.

UFW konfiguriert IPv4 und IPv6 parallel, sofern in `/etc/default/ufw` `IPV6=yes` gesetzt ist (Default auf Ubuntu 24.04 / Debian 12). Verifiziere das:

```
grep ^IPV6 /etc/default/ufw      # erwartet: IPV6=yes
ufw status verbose | grep -i v6  # bestätigt v6-Regeln
```

Wenn du keinen AAAA-Record setzt und IPv6 nicht nutzt, bleibt der v6-Stack ein potentieller blinder Fleck. Im Zweifel später beim Portscan **IPv4 und IPv6** prüfen.

Schritt 2 — Unbound installieren und konfigurieren

```
apt install -y unbound dns-root-data
```

Das `dns-root-data` -Paket installiert die Root-Hints (`/usr/share/dns/root.hints`) und den DNSSEC-Root-Trust-Author (`/var/lib/unbound/root.key`). Das Unbound-Paket von Ubuntu/Debian liefert ein Default-Snippet unter `/etc/unbound/unbound.conf.d/root-auto-trust-anchor-file.conf` mit, das `auto-trust-anchor-file` bereits setzt — **dupliziere diese Zeile NICHT in deiner eigenen Config, sonst startet Unbound nicht.**

Lege `/etc/unbound/unbound.conf.d/custom.conf` an:

```

server:
  # Nur auf Loopback lauschen, auf einem nicht-Standard-Port (NICHT 5353 – das ist
  # mDNS).
  interface: 127.0.0.1
  port: 5335
  do-ip4: yes
  do-ip6: no          # nur auf yes, wenn dein VPS funktionierendes IPv6 hat, dann
                     # echt geil, kann nochmal deutlich schneller sein
  do-udp: yes
  do-tcp: yes

  # Nur Anfragen von localhost akzeptieren
  access-control: 127.0.0.0/8 allow
  access-control: 0.0.0.0/0 refuse
  access-control: ::0/0 refuse

  hide-identity: yes
  hide-version: yes

  # Hardening
  harden-glue: yes
  harden-dnssec-stripped: yes
  harden-below-nxdomain: yes
  harden-referral-path: yes
  qname-minimisation: yes
  use-caps-for-id: no  # yes für 0x20-Randomisierung; bricht ein paar Sites

  # Der DNSSEC-Trust-Anchor wird vom Default-Snippet des Pakets gesetzt,
  # auto-trust-anchor-file hier NICHT erneut deklarieren.

  # Schutz gegen DNS-Rebinding / Antworten mit privaten Adressen.
  # Für eigene interne Zonen zusätzlich local-zone / private-domain konfigurieren.
  private-address: 10.0.0.0/8
  private-address: 172.16.0.0/12
  private-address: 192.168.0.0/16
  private-address: 169.254.0.0/16
  private-address: fd00::/8
  private-address: fe80::/10

  # Cache (auf VPS-RAM anpassen; das hier ist konservativ für ~1 GB)
  msg-cache-size: 64m
  rrset-cache-size: 128m
  msg-cache-slabs: 4
  rrset-cache-slabs: 4
  infra-cache-slabs: 4
  key-cache-slabs: 4

  # Performance
  prefetch: yes
  prefetch-key: yes
  num-threads: 2      # an deine vCPU-Anzahl anpassen
  so-rcvbuf: 1m

  unwanted-reply-threshold: 10000000
  edns-buffer-size: 1232 # vermeidet Fragmentierung auf den meisten Pfaden

remote-control:

```

```
# Aktiviert "unbound-control" (für Stats und Live-Befehle, später nützlich)
control-enable: yes
control-interface: 127.0.0.1
```

Einmalig die Control-Schlüssel erzeugen, dann validieren, starten, testen:

```
unbound-control-setup
unbound-checkconf
systemctl restart unbound
systemctl enable unbound

# Positiv-Test: gut signierte Domain → Antwort mit "ad"-Flag
dig @127.0.0.1 -p 5335 +short cloudflare.com
dig @127.0.0.1 -p 5335 +dnssec cloudflare.com | grep "flags:"

# Negativ-Test: bewusst kaputte DNSSEC-Signatur → MUSS SERVFAIL zurückgeben
dig @127.0.0.1 -p 5335 dnssec-failed.org | grep -E "status:|flags:"
```

Der zweite Befehl muss `flags: qr rd ra ad` zeigen — das `ad`-Flag heißt **Authenticated Data**: DNSSEC-Validierung erfolgreich. Der Negativ-Test mit `dnssec-failed.org` muss `status: SERVFAIL` zeigen — das beweist, dass Unbound kaputte Signaturen wirklich verwirft (und nicht einfach "alles OK" stempelt). Beides zusammen ist der echte Validierungs-Beleg.

Warum Port 5335 und nichts anderes

5335 ist der konventionelle Port für Unbound-hinter-AdGuard. **Nimm nicht 5353** — das ist der Well-Known-Port für mDNS (Avahi/Bonjour) und kollidiert mit allem, was `.local`-Auflösung macht.

Schritt 3 — Port 53 für AdGuard Home freiräumen

Erst kurz checken, ob das auf deinem System überhaupt nötig ist — auf Ubuntu 24.04 läuft `systemd-resolved` per Default, auf Debian 12 nicht zwingend:

```
systemctl status systemd-resolved --no-pager
resolvectl status 2>/dev/null | head -5
ss -tulpen | grep ':53'
```

Wenn `systemd-resolved` läuft und auf `127.0.0.53:53` (oder `127.0.0.54:53`) lauscht, geh weiter. Wenn nicht, kannst du diesen Schritt überspringen — Port 53 ist frei.

`systemd-resolved` benutzt seine internen Stub-Adressen `127.0.0.53` und `127.0.0.54` auf Port 53 (nicht zu verwechseln mit mDNS auf Port 5353). Damit AdGuard Home `0.0.0.0:53` binden kann (damit DoT auch von außen erreichbar ist — DoT teilt sich den DNS-Bind), muss der Stub-Listener weg.

```
mkdir -p /etc/systemd/resolved.conf.d
cat > /etc/systemd/resolved.conf.d/disable-stub.conf <<EOF
[Resolve]
DNSStubListener=no
EOF
ln -sf /run/systemd/resolve/resolv.conf /etc/resolv.conf
systemctl restart systemd-resolved
```

Das deaktiviert nur den *Stub-Listener* auf Port 53. `systemd-resolved` selbst läuft weiter — `/etc/resolv.conf` zeigt jetzt auf die echte Backing-File des Dienstes, damit der Server selbst weiterhin Namen auflösen kann (apt-Updates etc.).

Verifiziere, dass auf `:53` nichts mehr lauscht (TCP **und** UDP, denn DNS läuft auf beidem):

```
ss -tulpen | grep ':53'
```

Sollte leer sein.

Schritt 4 — AdGuard Home installieren

Offizieller Installer (es ist eine gute Angewohnheit, Skripte vor dem Ausführen kurz anzuschauen — `curl ... | sh` blind ausführen sollte man sich abgewöhnen):

```
curl -s -S -L https://raw.githubusercontent.com/AdguardTeam/AdGuardHome/master/scripts/install.sh | sh -s -- -v
```

Das lädt die Binary nach `/opt/AdGuardHome/`, installiert eine systemd-Unit und startet den Setup-Wizard auf Port 3000.

Setup-Wizard per SSH-Tunnel erreichen

Das Admin-UI nicht öffentlich exponieren. Von deinem lokalen Rechner aus:

```
ssh -L 3000:localhost:3000 mein-dns
```

Dann `http://localhost:3000` im Browser öffnen. Geh durch den Wizard:

- **Admin-Web-Interface:** Wizard läuft initial auf `0.0.0.0:3000` und kann beim ersten Run nicht direkt auf `127.0.0.1` umbinden, weil der laufende Prozess den Port schon belegt. Lass den Wizard fertiglaufen, dann editier `/opt/AdGuardHome/AdGuardHome.yaml` und ändere `http.address` auf `127.0.0.1:3000`, anschließend `systemctl restart AdGuardHome`. **Die Firewall blockt 3000 zwar, aber Defense-in-Depth: das Admin-UI hat keinen Grund, auf einer öffentlichen Schnittstelle zu lauschen.**
- **DNS-Server:** `0.0.0.0:53`. Die "Autofix"-Option im Wizard kümmert sich um den `systemd-resolved`-Konflikt, aber wenn du Schritt 3 schon erledigt hast, brauchst du sie nicht.

- **Admin-User/Passwort:** starkes, einmaliges Passwort (24+ Zeichen, zufällig). AGH speichert es bcrypt-gehasht. **In den Passwort-Manager packen.**

Schritt 5 — TLS-Zertifikat (Let's Encrypt)

```
apt install -y certbot
certbot certonly --standalone --non-interactive --agree-tos \
  --email du@example.com -d dns.example.com
```

Hostname und E-Mail durch deine ersetzen. Das nutzt die **HTTP-01-Challenge**: certbot bindet kurz Port 80 während Issuance/Renewal, beweist, dass dir der Hostname gehört, und holt das Zertifikat.

Die Cert-Dateien landen in:

```
/etc/letsencrypt/live/dns.example.com/fullchain.pem
/etc/letsencrypt/live/dns.example.com/privkey.pem
```

(Sind Symlinks auf versionierte Dateien in `/etc/letsencrypt/archive/`.)

AdGuard Home automatisch reloaden bei Renewal

Renewals laufen zweimal täglich per `certbot.timer`. Damit AGH neue Certs ohne manuellen Restart einliest:

```
mkdir -p /etc/letsencrypt/renewal-hooks/deploy
cat > /etc/letsencrypt/renewal-hooks/deploy/reload-agh.sh <<'EOF'
#!/bin/sh
systemctl reload AdGuardHome 2>/dev/null || systemctl restart AdGuardHome
EOF
chmod +x /etc/letsencrypt/renewal-hooks/deploy/reload-agh.sh
```

Verifizieren, dass Renewal end-to-end funktioniert, ohne tatsächlich auszustellen:

```
certbot renew --dry-run
```

Alternative: DNS-01-Challenge. Wenn dein Domain-Registrar eine API hat (Cloudflare, Hetzner DNS, deSEC, etc.), ist DNS-01 sauberer — kein Port 80, unterstützt Wildcards. Das passende `python3-certbot-dns-*`-Plugin und ein API-Token nutzen. Lohnt sich langfristig.

Schritt 6 — AdGuard Home konfigurieren

Zurück ins Admin-UI (über deinen SSH-Tunnel, `http://localhost:3000`):

Settings → DNS settings

- **Upstream DNS servers:** `127.0.0.1:5335` (und **nur** das). Damit zeigt AGH auf dein lokales Unbound.
- **Bootstrap DNS:** `1.1.1.1`, `9.9.9.9` — wird nur benutzt, um Upstream-Hostnames beim Start aufzulösen. Egal, wenn der Upstream eine lokale IP ist, aber AGH will hier was haben.
- **Upstream mode:** Load balancing (irrelevant bei nur einem Upstream).
- **DNS cache size:** **anlassen**, auf was Sinnvolles wie 256 MB stellen. Es gibt schlechten Rat im Internet, der dir sagt, du sollst AGHs Cache deaktivieren, weil Unbound auch cached — ignorier das. AGHs Cache spart einen Round-Trip für Wiederholungsanfragen und entlastet Unbound gratis.
- **Enable DNSSEC:** **aus**. Unbound validiert schon. Doppel-Validierung verschwendet nur CPU.
- **Rate limiting:** auf Default lassen (~20 Anfragen/Sek pro Client). Dein Endpunkt ist öffentlich, das ist relevant.

Settings → DNS blocklists (Filters)

Sinnvolles Starter-Set:

- AdGuard DNS filter (per Default an)
- AdAway Default Blocklist
- Dan Pollock's List
- OISD (Big) — groß, aber gut gepflegt, wenig False Positives

Übertreib es nicht. Mehr Listen = mehr Speicher und mehr False-Positive-Ärger. Du kannst später immer noch was nachlegen, wenn dir bestimmte Sachen durchrutschen.

Settings → Encryption settings

- **Enable encryption:** an
- **Server name:** `dns.example.com`
- **HTTPS port:** `443`
- **DNS-over-TLS port:** `853` (überspringen, falls du 853 in der Firewall nicht aufgemacht hast)
- **Certificate path:** `/etc/letsencrypt/live/dns.example.com/fullchain.pem`
- **Private key path:** `/etc/letsencrypt/live/dns.example.com/privkey.pem`

AGH muss die Cert-Dateien lesen können. Die Let's-Encrypt-Verzeichnisse `/etc/letsencrypt/live/` und `/etc/letsencrypt/archive/` sind 0700 und root-owned. Das einfachste Setup: AGH läuft als root (Default vom Installer). Single-Purpose-Box, Single-Tenant — vertretbar. Wenn du AGH nicht als root willst, leg eine Gruppe an, mach den AGH-User Mitglied, und `chmod 0750` die LE-Verzeichnisse mit der Gruppe.

Nach dem Speichern startet AGH neu und dein DoH/DoT-Endpunkt ist live unter:

- **DoH:** <https://dns.example.com/dns-query>
- **DoT:** dns.example.com auf Port 853

Achtung: Der Endpunkt ist ab jetzt **öffentlich nutzbar** — jeder im Internet kann ihn als DNS-Resolver verwenden. Bevor du das Setup als “fertig” betrachtest, mach Schritt 6.5.

Schritt 6.5 — Zugriff einschränken (Pflicht, nicht optional)

Rate Limiting hilft gegen Lasteffekte, ist aber **keine Authentifizierung**. Wenn du einen privaten Resolver willst, musst du Fremde rauswerfen. AdGuard Home unterstützt dafür eine Allowlist mit IPs, CIDRs und sogenannten **ClientIDs**. Mindestens eines davon brauchst du.

Variante A — Allowlist nach IP (am einfachsten, wenn du eine feste Heim-IP hast)

AGH-UI → **Settings** → **DNS settings** → **Access settings** → **Allowed clients**, dort deine Heim-IP / dein WG-Netz / dein VPN-Netz eintragen. Beispiel-Inhalt:

```
203.0.113.10      # Heim-IP
2001:db8:abc::/64 # IPv6-Heim-Präfix, falls vorhanden
10.10.0.0/24      # WireGuard-Netz, falls du eines hast
```

Limitierung: Heim-IPs ändern sich (Zwangstrennung, neuer Router, Mobilfunk). Für **mobile Geräte ist IP-basiert ungeeignet** — die wechseln IPs ständig.

Variante B — ClientIDs (für mobile Geräte und überall, wo die IP nicht stabil ist)

ClientIDs sind ein AGH-Feature, bei dem jeder Client einen geheimen Bezeichner im DoH-Pfad bzw. DoT-Hostnamen mitschickt. Beispiel:

- **DoH-URL pro Gerät:** <https://dns.example.com/dns-query/cedric-iphone>
- **DoT-Hostname pro Gerät:** cedric-iphone.dns.example.com

In **Settings** → **DNS settings** → **Access settings** → **Allowed clients** trägst du nur die ClientIDs ein, einen pro Zeile:

```
cedric-iphone
cedric-laptop
familien-tablet
```

Vergib pro Gerät eine eigene, schwer ratbare ID (mindestens 16+ Zufallszeichen — ClientIDs landen in Logs/Cert-SNI, sind also nicht “Passwörter”, aber sollten nicht trivial ratbar sein). Beispiel-Generator:

```
head -c 16 /dev/urandom | base32 | tr A-Z a-z | tr -d '='
```

Auf den Geräten dann **diese URL** statt `https://dns.example.com/dns-query` eintragen. Für DoT auf Android: **diesen Hostnamen** statt `dns.example.com` als “Privates DNS” verwenden — Android hängt die Subdomain automatisch in SNI mit, AGH liest sie aus.

Für DoT-Subdomain-ClientIDs brauchst du entweder ein Wildcard-Cert (`*.dns.example.com`) oder du listest die genauen Hostnamen im Cert. Mit Let’s Encrypt + DNS-01-Challenge bekommst du Wildcard kostenlos. Mit HTTP-01 musst du jeden Hostnamen einzeln signieren lassen. **DoH-ClientIDs (Pfad-basiert) brauchen kein Wildcard-Cert** — also der einfachere Weg.

Verifizieren, dass die Sperre wirkt

Nach Aktivierung der Allowlist von einem **nicht zugelassenen** Client testen:

```
kdig +https @dns.example.com example.com
```

Erwartung: Refused / Timeout / leere Antwort. Wenn du eine normale Antwort kriegst, ist die Allowlist nicht aktiv.

Schritt 6.6 — Admin-UI absichern (auch Pflicht)

Sobald TLS in AdGuard aktiv ist, läuft das Admin-UI auf demselben HTTPS-Listener wie DoH. Heißt: `https://dns.example.com/` zeigt von überall im Internet die AGH-Login-Seite. Der Login ist rate-limited und das Passwort bcrypt-gehasht — aber das ist trotzdem ein öffentlicher Loginprompt mit Brute-Force-Oberfläche und potentiellen Auth-Bugs in der Vergangenheit.

Drei sinnvolle Wege:

1. Caddy als Reverse Proxy davor (empfohlen). Caddy übernimmt TLS und lässt nur `/dns-query...` durch:

```
apt install -y caddy
```

`/etc/caddy/Caddyfile` so anlegen — vorher AGH-TLS-Listener auf einen lokalen Port verlegen, damit Caddy 443 hat (`/opt/AdGuardHome/AdGuardHome.yaml` : `tls.port_https: 8443` , dann `systemctl restart AdGuardHome`):

```

dns.example.com {
  @doh {
    path /dns-query*
    method GET POST
  }
  handle @doh {
    reverse_proxy 127.0.0.1:8443 {
      transport http {
        tls
        tls_server_name dns.example.com
      }
      header_up X-Real-IP {remote_host}
    }
  }
  handle {
    respond "Not found" 404
  }
}

```

Wenn du Caddy nimmst, brauchst du Certbot für diesen Host nicht mehr — Caddy holt und erneuert das Zertifikat selbst. Falls Certbot vorher schon lief: `systemctl disable --now certbot.timer` und ggf. `certbot delete --cert-name dns.example.com`.

`trusted_proxies` sollte AGH wissen (`/opt/AdGuardHome/AdGuardHome.yaml` → `dns.trusted_proxies: ['127.0.0.1']`), damit `X-Real-IP` korrekt als Client-IP für Allowlist und Logging gewertet wird.

DoT (Port 853) terminiert weiterhin AdGuard direkt — Caddy proxyt nur HTTPS.

2. AGH-Admin-UI auf Loopback + nur per SSH-Tunnel erreichen. Wenn du keinen Reverse Proxy willst: `http.address: 127.0.0.1:3000` setzen und das Web-UI ausschließlich per SSH-Tunnel öffnen:

```
ssh -L 3000:localhost:3000 mein-dns
```

Reduziert die Angriffsfläche auf 3000 weiterhin nicht (Port 443 zeigt weiter die Login-Seite), löst das Problem also nur halb. Reicht für Hobby-Setups, ersetzt aber Variante 1 nicht.

3. WireGuard / Tailscale davor. Admin-UI nur über das interne Mesh erreichbar machen, Port 3000 nicht öffentlich exponieren. Sauber, wenn du sowieso ein VPN nutzt.

Schritt 7 — Alles verifizieren

Von deinem lokalen Rechner aus

```
# DoH Wire-Format (das nutzen echte Clients)
kdig +https @dns.example.com cloudflare.com

# DoT
kdig +tls @dns.example.com cloudflare.com

# Beide müssen Antworten mit dem "ad"-Flag liefern:
# ;; Flags: qr rd ra ad
```

`kdig` ist Teil des Pakets `knot-dnsutils` auf Debian/Ubuntu.

Bestätigen, dass du kein Open Resolver bist

Von einem Rechner **außerhalb** deines VPS:

```
# Plain DNS auf 53 muss timeout/refused liefern – niemals eine echte Antwort
dig @DEIN.VPS.IP google.com
dig @DEIN.VPS.IP google.com +tcp

# Externer Portscan, IPv4 – nur 22, 80, 443, 853 dürfen offen sein
nmap -Pn -p 22,53,80,443,853,3000 DEIN.VPS.IP

# Wenn dein VPS auch IPv6 hat, dasselbe für v6
nmap -6 -Pn -p 22,53,80,443,853,3000 DEIN:VPS:IPv6::1
```

Wenn einer der `dig`-Befehle eine Antwort liefert, **stopp und fix die Firewall**, bevor du weitermachst. Open Resolvers werden innerhalb von Stunden für Amplification-Angriffe rekrutiert.

Port 80 darf in nmap als `closed` / `filtered` erscheinen — Certbot bindet ihn nur während der Renewal-Challenge. Wichtig ist nur, dass UFW ihn erlaubt, damit Let's Encrypt durchkommt.

Auch lohnt sich ein Check auf <https://openresolver.com/> — VPS-IP eintragen, sollte "not an open resolver" zurückgeben.

Bestätigen, dass der DoH/DoT-Endpunkt nicht öffentlich nutzbar ist

`openresolver.com` testet nur Port 53. Den DoH/DoT-Missbrauch testest du selbst, von einem **nicht in der Allowlist eingetragenen** Client:

```
kdig +https @dns.example.com example.com
kdig +tls @dns.example.com example.com
```

Erwartung mit aktiver Allowlist: keine normale Antwort (Refused/leer). Wenn du Antworten kriegst, ist die Allowlist nicht aktiv und dein Resolver ist trotz DoH/DoT-Verschlüsselung öffentlich nutzbar.

Bestätigen, dass der Filter greift

```
kdig +https @dns.example.com doubleclick.net
```

Erwartung: blockierte Antwort. Je nach AdGuard-Blocking-Mode bekommst du `0.0.0.0`, `NXDOMAIN` oder eine AdGuard-spezifische Block-Antwort — der genaue Wert variiert, wichtig ist nur, dass es **nicht** die echte IP ist.

Client-Setup

DoH und DoT sind unterschiedliche Protokolle. Pro Gerät eines auswählen — beide tun den gleichen Job.

Client	Protokoll	Was du einträgst
Firefox	DoH	Einstellungen → Datenschutz → DNS über HTTPS → Maximaler Schutz → Benutzerdefiniert: https://dns.example.com/dns-query
Chrome / Edge / Brave	DoH	Einstellungen → Datenschutz → Sicheres DNS → Benutzerdefiniert: https://dns.example.com/dns-query
Windows 11	DoH	Einstellungen → Netzwerk → Adapter → DNS-Serverzuweisung → Manuell → Verschlüsseltes DNS → DNS over HTTPS, Mit: Benutzerdefiniert → URL oben. (UI ist je nach Build wackelig — Alternative siehe PowerShell-Block unten.)
Android 9+	siehe Mobile unten	Native “Privates DNS” macht nur DoT , ohne Pfad. Mit ClientID-Allowlist nicht direkt nutzbar — Mobile-Strategie ist eigener Block.
iOS / macOS	DoH oder DoT	.mobileconfig -Profil installieren. AGH-Admin-UI → Setup Guide → “DNS Privacy” generiert eines. Über Einstellungen → Allgemein → VPN & Geräteverwaltung installieren.
Linux Desktop	DoH	Einfachster Weg: pro Browser konfigurieren. Systemweit braucht es einen lokalen Stub wie dnscrypt-proxy oder cloudflared --proxy-dns .
Router (OpenWrt etc.)	DoH	Paket https-dns-proxy , auf deine DoH-URL zeigen lassen. Jedes Gerät im LAN kriegt automatisch gefiltertes DNS.

Die generelle Regel:

- **DoH** → komplette URL mit [/dns-query](#) (oder [/dns-query/<clientid>](#), falls du Allowlist per ClientID benutzt)

- **DoT** → nur der Hostname (Port 853 ist implizit; bei ClientID-Allowlist `<clientid>.dns.example.com` , nur mit Wildcard-Cert)

Mobile (Android, iPhone unterwegs)

Wenn du Allowlist sauber durchziehen willst, ist Mobile der schwierigste Teil — und du solltest **DoT-Port 853 nicht offen lassen, nur weil Android nichts anderes spricht**. Drei Wege, in der Reihenfolge wie ich sie empfehle:

1 — VPN nach Hause / zum VPS (klar empfohlen). WireGuard oder Tailscale auf dem VPS oder einem Heim-Gerät. Mobile geht über den VPN-Tunnel, nutzt deinen Resolver als interne IP. Vorteile: - Port 853 kann komplett zu — Resolver muss nicht öffentlich erreichbar sein - Funktioniert mit beliebigen DNS-Methoden (auch plain DNS auf Loopback im Tunnel) - Bonus: dein Mobile-Traffic ist unterwegs verschlüsselt - Nachteil: dauerhafter Tunnel = Akku-Impact, je nach VPN-Client gering bis spürbar

WireGuard auf dem VPS ist in 10 Minuten gemacht; Tailscale kostet sich keine Mühe (`curl -fsSL https://tailscale.com/install.sh | sh` , beitreten, fertig). Auf Android pushed du dann den DNS-Server `100.x.x.x` (Tailscale) oder `10.x.x.x` (WireGuard) — Tunnel auf, DNS geht durch deinen Resolver.

2 — DoH-App mit Custom URL (wenn VPN keine Option ist). Apps wie **Intra** (Google/Jigsaw, Open Source), **RethinkDNS** oder **AdGuard für Android** unterstützen Custom DoH URLs inklusive ClientID-Pfad. Caveats: - Diese Apps nutzen die Android-VPN-API → kein zweiter VPN-Client (z. B. NordVPN) gleichzeitig möglich - Erste Einrichtung erfordert das Erteilen der VPN-Permission im Systemdialog - Battery-Optimierung kann den Service killen → in den App-Einstellungen “nicht optimiert” setzen

3 — Wildcard-Cert + DoT-Subdomain (für Native “Privates DNS”). DNS-01-Challenge mit einem Provider, dessen API certbot kennt (Cloudflare, Hetzner DNS, deSEC, etc.). Cert auf `*.dns.example.com` ausstellen, AGH ClientID per Subdomain (`mobile-xxx.dns.example.com`). Dann läuft Androids native “Privates DNS”-Funktion ohne App. Aufwändiger, aber sauberste UX langfristig.

Was du nicht tun solltest: Port 853 für die Welt offen lassen, nur damit Mobile-Geräte ohne ClientID DoT machen können. Das öffnet deinen Resolver wieder für jeden Internetnutzer. Wenn dir VPN zu nervig ist und du Wildcard-Cert nicht magst, ist DoH-via-App-mit-ClientID der goldene Mittelweg.

Windows 11 zuverlässig per PowerShell

Die Windows-UI für DoH ist je nach Build mal vorhanden, mal kaputt, mal an eine “bekannte” DoH-Server-IP gebunden. Sauberer ist `Add-DnsClientDohServerAddress` als Admin-PowerShell:

```
Add-DnsClientDohServerAddress `
-ServerAddress DEIN.VPS.IP `
-DohTemplate "https://dns.example.com/dns-query" `
-AllowFallbackToUdp $False `
-AutoUpgrade $True
```

Anschließend pro Adapter den DNS-Server auf `DEIN.VPS.IP` setzen — Windows nutzt dann automatisch die DoH-Vorlage. **AllowFallbackToUdp \$False ist wichtig:** sonst fällt Windows bei Problemen still auf unverschlüsseltes UDP/53 zurück.

Den Server betreiben

Wo schauen

- **AdGuard-Home-Dashboard:** Anfragen, Blocked-Counts, Top-Clients, Latenz, Activity. Über den SSH-Tunnel:

```
ssh -L 3000:localhost:3000 mein-dns
```

Dann `http://localhost:3000`.

- **Unbound-Stats** (nur CLI — Unbound hat kein Web-UI):

```
ssh mein-dns 'unbound-control stats_noreset'
```

Nützliche Keys: `total.num.queries`, `total.num.cachehits`, `mem.cache.rrset`, `time.up`.

Updates

- **Unbound und OS:** `apt update && apt upgrade` — monatlich oder unattended-upgrades aktivieren.
- **AdGuard Home:** hat einen eingebauten Updater. AGH-UI → General → “Check for updates” oder Auto-Updates an.
- **Filter-Listen:** AGH refreshed automatisch (Default alle 24h).
- **Cert-Renewal:** automatisch über `certbot.timer`. Ab und zu mal checken:

```
systemctl status certbot.timer
certbot certificates
```

Backups

Die ganze Config ist klein und backupwürdig:

- `/etc/unbound/unbound.conf.d/custom.conf`
- `/opt/AdGuardHome/AdGuardHome.yaml` (enthält deinen bcrypt-Admin-Hash und TLS-Settings)
- `/etc/letsencrypt/` (das Cert-Verzeichnis)

- `/etc/ssh/sshd_config.d/99-hardening.conf`

Wenn du einen One-Liner-Backup-Befehl willst:

```
tar czf /root/dns-backup-$(date +%F).tar.gz \
  /etc/unbound /opt/AdGuardHome/AdGuardHome.yaml \
  /etc/letsencrypt /etc/ssh/sshd_config.d
```

Dann das Tarball regelmäßig per `scp` von der VPS runterziehen. Die einzige Kopie nicht auf der Box lassen, die du backuppst.

Sicherheits-Tipps

Nicht verhandelbar

1. **Erst Firewall, dann Dienste.** UFW als erstes einrichten, Default-Deny inbound, nur die Ports, die du wirklich brauchst. Nach jeder Config-Änderung mit externem Portscan verifizieren — IPv4 und IPv6.
2. **SSH nur per Key.** Passwort-Auth aus, Root-Login nur per Key. Privaten Key an zwei Stellen backuppen. Wenn du es härter willst: separater sudo-User, `PermitRootLogin no`, Key nur für den User. Root-per-Key ist für eine Single-Purpose-Box vertretbar, aber der Blast-Radius ist größer.
3. **Allowlist (Schritt 6.5) ist Pflicht.** Ohne sie ist dein Resolver für jeden im Internet nutzbar.
4. **Admin-UI nicht öffentlich (Schritt 6.6).** Reverse Proxy oder VPN-only — AGHs Login auf <https://dns.example.com/> ist eine vermeidbare Angriffsfläche.
5. **Starkes, einmaliges Admin-Passwort für AGH.** Auch hinter einem Reverse Proxy.
6. **Cert-Renewal überwachen.** Ein gescheitertes Renewal killt DoH leise nach 90 Tagen. Nach dem Setup einmal `certbot renew --dry-run` laufen lassen, um zu bestätigen, dass der Pfad funktioniert. Dann entweder die Expiry-Notice-Mails von Let's Encrypt abonnieren oder eigenen Check einrichten.
7. **AGH und Unbound aktuell halten.** Beide hatten über die Jahre sicherheitsrelevante Fixes.

TLS-Verantwortung sauber trennen

Du hast drei Varianten für TLS — entscheid dich für **eine**, sonst gibt's Port-80-/Cert-Konflikte:

- **AdGuard terminiert TLS selbst** → Certbot standalone (HTTP-01, Port 80 zeitweise frei) oder DNS-01.
- **Caddy davor (Schritt 6.6)** → Caddy holt und erneuert das Cert automatisch; **kein Certbot** für diesen Host. Vorher laufenden Certbot-Timer abschalten.
- **DNS-01 mit certbot** → kein Port 80 nötig, Wildcards möglich (gut für DoT-ClientIDs).

SSH

- SSH von Port 22 wegmovieren reduziert Log-Noise (verändert nichts an der echten Sicherheit, hält aber die `auth.log` sauber). UFW entsprechend anpassen.
- `fail2ban` installieren, um Brute-Force-Versuche automatisch zu bannen:

```
apt install -y fail2ban
systemctl enable --now fail2ban
```

- Für den Paranoia-Modus: Port 22 komplett aus der öffentlichen Firewall raus, SSH nur über WireGuard erlauben.

Rate Limiting

Dein DoH-Endpunkt ist öffentlich. AGH macht Rate Limiting pro Client; lass es an. Wenn du komischen Traffic im AGH-Dashboard siehst (viele Anfragen aus einer Quelle), zieh enger an (Settings → DNS settings → Rate limit) oder block die IP an der Firewall.

Privacy beim Query Log

AGH loggt per Default jede Anfrage mit Timestamp + Client-IP. Entscheid dich für eine Aufbewahrungsdauer:

- AGH-UI → Settings → General → Query log
- 24 Stunden reichen für die meisten privaten Setups
- Wenn mehrere Leute deinen Resolver nutzen und du nicht wissen willst, was sie nachschauen, schalt das Log komplett aus oder anonymisier Client-IPs (Settings → DNS settings → "Anonymize client IP")

Was du im Hinterkopf haben solltest

Operativ

- **Deine VPS-IP macht jetzt deine ganzen rekursiven Lookups.** Sites mit aggressivem Geo-DNS (Netflix, regionale CDNs, Akamai-fronted Services) routen nach VPS-Standort, nicht nach deinem. Meistens egal, manchmal kriegst du eine "falsche" CDN-Edge — etwas langsamere Downloads von genau dem Service.
- **Cold-Cache-Latenz.** Rechne mit 20–80 ms extra auf nicht-gecachte Lookups gegenüber 1.1.1.1 / 8.8.8.8. Mit Unbounds Prefetch bleiben populäre Domains warm und das normale Surfen fühlt sich identisch an.
- **Du bist jetzt deine eigene Abhängigkeit.** Wenn der VPS down ist, ist dein DNS down. Ein Fallback ist ein **Trade-off, kein No-Brainer**: hängst du als Backup 1.1.1.1 rein, gehen Anfragen bei jedem Timeout / Wackler / Wifi-Wechsel still an Cloudflare statt an deinen Resolver — Filter weg, Privacy weg, ohne dass du es merkst. Sauberer ist ein **zweiter eigener Resolver** auf einem zweiten VPS (anderer Anbieter, andere Region)

als Backup, oder ein lokaler Notfallresolver im Heimnetz. Wenn du trotzdem einen öffentlichen Fallback einträgst, sei dir des Trade-offs bewusst.

- **DoT teilt sich den DNS-Bind.** Der DoT-Listener von AGH nutzt dieselbe `bind_hosts` wie plain DNS. Wenn du plain DNS aus Sicherheitsgründen an `127.0.0.1` bindest, geht DoT auch nur Loopback und ist nicht erreichbar. Bind an `0.0.0.0` und vertrau drauf, dass die Firewall plain `:53` blockt.

Performance

- **RAM:** Unbound-Cache + AGH-Cache + Filter-Listen $\approx$ 512 MB Minimum, 1 GB komfortabel. OISD plus den Rest des Starter-Sets sind etwa $\sim$ 150 MB extra in AGH.
- **CPU:** 1 vCPU reicht für Privatgebrauch. Unter heftigem Missbrauch wäre mehr besser, aber dann zählt der Rate Limiter mehr als die CPU.
- **IPv6:** `do-ip6: yes` in Unbound nur an, wenn dein VPS tatsächlich funktionierendes IPv6 outbound hat. Kaputtes IPv6 = langsame Lookups, weil Unbound auf AAAA-Pfaden in Timeouts läuft.

Häufige Probleme und Fixes

Unbound startet nicht: “trust anchor presented twice”

Du hast `auto-trust-anchor-file` in deine custom Config gepackt. Raus damit — das Default-Snippet des Pakets setzt das schon. Deine Custom-Config darf diese Direktive *nicht* enthalten.

AGH startet nicht: “bind: address already in use” auf :53

Etwas anderes belegt Port 53. Auf Ubuntu/Debian ist das fast immer `systemd-resolved`. Siehe **Schritt 3** — `DNSStubListener=no` setzen und `systemd-resolved` neu starten.

DoH funktioniert lokal, aber nicht von außen

- Firewall checken: `ufw status`. Ist 443 offen?
- Checken, ob AGH wirklich auf `*:443` lauscht:

```
ss -tlnp | grep ':443'
```

- Cert-Validität mit dem richtigen Hostname checken:

```
echo | openssl s_client -connect dns.example.com:443 -servername dns.example.com 2>/dev/null \
| openssl x509 -noout -subject -issuer -dates
```

- Sicherstellen, dass dein A-Record auch wirklich auf den VPS zeigt:

```
dig +short dns.example.com @1.1.1.1
```

Cert-Renewal scheitert

`certbot renew --dry-run` laufen lassen und den Fehler genau lesen. Häufige Ursachen:

- **Port 80 blockiert** in der Firewall, beim VPS-Anbieter oder in einer Security Group. HTTP-01 braucht Port 80 erreichbar von den Validatoren von Let's Encrypt aus.
- **Etwas Anderes auf Port 80.** Certbot muss ihn während Renewal binden — kein nginx/caddy darf ihn belegen. (Wenn du einen Webserver betreibst, hat certbot Plugins für nginx/apache, die den Port nicht freigeben müssen.)
- **DNS löst nicht auf.** Der Hostname muss öffentlich auf deinen VPS zeigen.

`dig @DEIN.VPS.IP` von außen liefert Antworten

Du hast einen Open Resolver. **Sofort die Firewall fixen.** Open Resolvers werden für Amplification-Angriffe missbraucht; manche VPS-Anbieter null-routen oder kündigen dich.

```
ufw status                                # bestätigen, dass 53 nicht erlaubt ist
ss -tulpen | grep ':53'                  # prüfen, was auf 53 lauscht (TCP + UDP)
```

Die Kombi aus UFW-blockt-Inbound-53 plus AGH/Unbound-bedienen-53-nur-über-die-firewallte-Schnittstelle ist der sichere Zustand.

`unbound-control` sagt “permission denied” oder “could not connect”

`remote-control` ist nicht aktiv oder die Schlüssel fehlen. Fix:

```
unbound-control-setup
systemctl restart unbound
unbound-control stats_noreset
```

Der `remote-control` -Block aus Schritt 2 muss in deiner Config stehen.

Fremde tauchen im AGH-Query-Log auf

Allowlist (Schritt 6.5) ist nicht aktiv oder leer. Bei aktiver Allowlist mit ClientIDs auch prüfen, dass alle eigenen Geräte den ClientID-Pfad / -Hostname benutzen — bare <https://dns.example.com/dns-query> ohne ClientID wird sonst gerefused, und Geräte landen still im Fallback.

Was du NICHT tun solltest

- **Plain DNS (Port 53) nicht öffentlich exponieren.** Niemals. Auch nicht mit Rate Limit. Es ist ein Amplification-Reflektor.
- **AGHs Cache nicht deaktivieren.** Häufiger schlechter Rat in Foren. Anlassen ist schneller *und* schonender für Unbound.
- **Keine zwei DNSSEC-Validatoren betreiben.** Validierung in Unbound, in AGH aus. Sonst nur CPU-Verschwendung.

- **Admin-UI auf Port 3000 nicht öffentlich exponieren.** Das Web-UI hatte schon Auth-Bugs in der Vergangenheit. SSH-tunneln oder per Reverse Proxy mit striktem Path-Matching.
 - **Den Firewall-Schritt nicht “erstmal später” machen.** Stunden eines offenen rekursiven Resolvers auf der öffentlichen IP reichen, damit jemand ihn findet.
-

Kosten und Zeit

- **VPS:** ~3–7 €/Monat für eine 1-vCPU/1–2-GB-Box. Manche Anbieter sind günstiger (Hetzner CX22, Netcup VPS 200 G11).
 - **Domain:** ~10 €/Jahr für `.com` / `.de` / `.xyz`.
 - **Setup-Zeit:** 30–60 Minuten beim ersten Mal, 15 Minuten, wenn du es schon gemacht hast.
 - **Laufender Aufwand:** ~10 Min/Monat für Updates und einen Blick aufs Dashboard. Die meisten Monate vergisst du, dass es überhaupt existiert.
-

Schnellreferenz: Ports und Dateien

Komponente	Lauscht auf	Notizen
SSH	0.0.0.0:22	nur Key
Reverse Proxy (Caddy, optional)	0.0.0.0:443	terminiert TLS, lässt nur <code>/dns-query*</code> durch
AGH DoH	127.0.0.1:8443 (hinter Caddy) oder 0.0.0.0:443 (ohne Proxy)	öffentlich; Allowlist Pflicht
AGH DoT	0.0.0.0:853 (firewall-zu per Default)	nur öffnen, wenn du Wildcard-Cert + Subdomain-ClientIDs einsetzt
AGH plain DNS	0.0.0.0:53	UFW-geblockt; nach Setup ggf. komplett deaktivieren
AGH Admin-UI	127.0.0.1:3000	nach <code>http.address</code> -Anpassung; nur SSH-Tunnel
Unbound rekursiv	127.0.0.1:5335	nur Loopback
Unbound remote-control	127.0.0.1:8953 (Default)	nur <code>unbound-control</code> , nur Loopback
HTTP (LE Renewal)	0.0.0.0:80	Listener nur on-demand während Renewal (entfällt mit Caddy oder DNS-01)

Datei	Zweck
<code>/etc/unbound/unbound.conf.d/custom.conf</code>	Unbound-Config
<code>/var/lib/unbound/root.key</code>	DNSSEC-Root-Trust-Anchor
<code>/opt/AdGuardHome/AdGuardHome.yaml</code>	AGH-Config (enthält Admin-Hash + TLS-Settings)
<code>/etc/letsencrypt/live/dns.example.com/</code>	TLS-Cert + Key
<code>/etc/systemd/resolved.conf.d/disable-stub.conf</code>	systemd-resolved-Port-53-Fix
<code>/etc/ssh/sshd_config.d/99-hardening.conf</code>	SSH-Hardening
<code>/etc/caddy/Caddyfile</code>	Optionaler Reverse Proxy (Schritt 6.6)

Die sechs Dinge, die wirklich zählen

1. **Erst Firewall, dann Dienste.** Default-Deny inbound, nur erlauben, was du brauchst. IPv4 *und* IPv6. Port 53 ist niemals öffentlich.
2. **Allowlist für DoH/DoT ist Pflicht.** Ohne sie ist der Resolver für jeden im Internet nutzbar — verschlüsselt heißt nicht privat.
3. **Admin-UI nicht öffentlich.** Reverse Proxy oder VPN-only. <https://dns.example.com/> darf keinen Loginprompt zeigen.
4. **Ein DNSSEC-Validator.** Unbound validiert, AGH nicht. Mit dnssec-failed.org testen, dass kaputte Signaturen wirklich verworfen werden.
5. **AGHs Cache anlassen.** Gratis-Latenz-Win.
6. **Cert-Renewal ist der stille Killer.** Einmal `certbot renew --dry-run` (oder Caddy-Equivalent) bestätigen, dann 89 Tage vergessen.

Alles andere sind Details.

Bis zum nächsten Mal im Cyberspace.