Inhaltsverzeichnis

Projekt: PiHole

Pi-hole (Vorstellung)

(R) Quelle https://de.wikipedia.org/

Pi-hole	
 Admin-Schnittstelle von Pi-hole	
Basisdaten	
Erscheinungsjahr	15. Juni 2015
Aktuelle Version	5.18.3[1] (5. Juli 2024)
Programmiersprache	Python
Lizenz	European Union Public Licence
pi-hole.net	

Pi-hole ist eine freie Software mit der Funktion eines Tracking- und Werbeblockers sowie eines optionalen DHCP-Servers. Pi-hole basiert auf einem Linux-System und ist für den Einsatz auf Kleinstcomputern im Sinne eines eingebetteten Systems entwickelt worden. Verbreitet ist der Einsatz auf Computern der Raspberry-Pi-Serie. Die Software wird als DNS-Server in ein bestehendes Netzwerk integriert und steht damit allen Geräten im Netzwerk zur Verfügung, deren DNS-Einstellungen sich konfigurieren lassen.

Inhaltsverzeichnis

Funktionsweise

Bearbeiten | Quelltext bearbeiten]

Das System mit der Pi-hole-Software wird als DNS-Server in ein bestehendes, zumeist kleineres, privates Rechnernetz eingebunden. Es übernimmt damit die Aufgabe, Domainanfragen der verbundenen Clients aufzulösen und in IP-Adressen umzuwandeln. Auf der Basis von Ausschlusslisten von bekannten Werbe- oder Trackingdomains und benutzerdefinierten Ausschlusslisten werden Anfragen entweder an konfigurierbare andere DNS-Server weitergeleitet oder, falls eine angefragte Domain in einer Ausschlussliste existieren sollte, eine technisch unbrauchbare IP-Adresse an den Client ausgeliefert (sog. DNS sinkhole). Durch die Übermittlung einer unbrauchbaren IP-Adresse an den Client kann dieser auf die angefragte Domain nicht zugreifen und folglich Werbung und/oder Tracking-Inhalte nicht abrufen.

Da in einem kleinen Netzwerk die Clients neben ihrer eigenen, lokalen IP-Adresse auch die Adresse des abzufragenden DNS-Servers vom DHCP-Server erhalten, bestehen drei unterschiedliche Wege, auf denen ein *Pi-hole* in ein bestehendes Netzwerk integriert werden kann:

1. Umkonfiguration des DHCP-Servers (in kleinen privaten Netzwerken wird diese Aufgabe normalerweise vom Router übernommen), so dass alle Clients als Standard-DNS-Server auf das Gerät mit der Pi-hole-Software verwiesen werden,
2. Abschaltung des bisherigen DHCP-Servers und Aktivierung des in der Pi-hole-Software integrierten DHCP-Servers, der in der Folge als Standard-DNS-Server auf sich selbst verweist oder
3. die individuelle DNS-Konfiguration einzelner Geräte innerhalb des lokalen Netzwerks.

Durch diese Funktionsweise ist es grundsätzlich und in der Standardeinstellung möglich, über die Benutzeroberfläche sämtliche Internetseiten, die von im Netzwerk befindlichen Geräten aufgerufen werden, nachzuverfolgen.[2]

Hardwareanforderungen

Bearbeiten | Quelltext bearbeiten]

Die Hardwareanforderungen sind gering. Wenn eine reine WLAN-Anbindung ausreichend ist, kann für den Betrieb in einem privaten Netzwerk ein Raspberry Pi zero eingesetzt werden.[4]

Technische Grenzen

Bearbeiten | Quelltext bearbeiten]

Eine Filterung erfolgt nur, soweit dies über Ausschlusslisten vordefiniert ist. Die Pi-hole-Software enthält keine Funktionen, welche angeforderte Inhalte inhaltlich überprüft.

Des Weiteren erfolgt eine Filterung nur auf Basis von Domainnamen. Werbung oder Tracking-Code, der clientseitig bereits durch den Zugriff auf konkrete IPs abgerufen wird und daher keine DNS-Anfrage benötigt, kann von der Pi-hole-Software nicht beeinflusst werden.

Pi-hole: Einrichtung und Konfiguration mit Fritz!Box – AdBlocker Teil1

https://www.kuketz-blog.de/pi-hole-einrichtung-und-konfiguration-mit-fritzbox-adblocker-teil1/
https://www.kuketz-blog.de/pi-hole-einrichtung-und-konfiguration-mit-unbound-adblocker-teil2/

1. Werbe- und trackerfrei Pi-hole mit Fritz!Box

Mittlerweile dürfte jeder den Einplatinencomputer Raspberry Pi kennen. Mit dem kleinen Computer im Scheckkartenformat lassen sich verschiedene Projekte realisieren. Eines davon ist der Pi-hole, mit dem sich Werbung und Tracker auf allen Geräten im (Heim-)Netzwerk ausblenden/blockieren lassen.

Der vorliegende Artikel bildet den Auftakt zu einer Artikelserie, in der verschiedene AdBlocker-Lösungen vorgestellt werden. Wir beginnen mit Pi-hole, der in Kombination mit einer Fritz!Box, einer weit verbreiteten Routerlösung von AVM, konfiguriert wird. Der Fokus liegt dabei auf einer einfachen Konfiguration, die leicht aktualisiert werden kann und sich an Anfänger und Fortgeschrittene richtet.

2. Was wir erreichen wollen

Dies ist eine Schritt-für-Schritt-Anleitung von der ersten Konfiguration eines Raspberry Pi bis zu dem Punkt, an dem Pi-hole alle DNS-Anfragen aus unserem Netzwerk beantwortet. Ziel ist es, ausgehende DNS-Anfragen an Werbe- und Tracking-Domains zu blockieren.

Wer direkt mit der Installation beginnen möchte, benötigt einen Raspberry Pi mit mindestens 512 MB RAM und eine (micro) SD-Karte mit mindestens 4 GB Speicher. Dann kann es direkt mit Punkt 4 weitergehen. Wer sich für die technischen Hintergründe und den geplanten Aufbau interessiert, sollte zuerst weiterlesen.

2.1 Konzept | Technischer Hintergrund

Am Beispiel von In-App-Werbung möchte ich zunächst erläutern, wie Pi-hole technisch funktioniert. Nehmen wir an, ein App-Entwickler hat ein Werbemodul in seine App integriert. Bei jedem Start der App oder auch während der Laufzeit kontaktiert die App bzw. das integrierte Modul die Adresse:

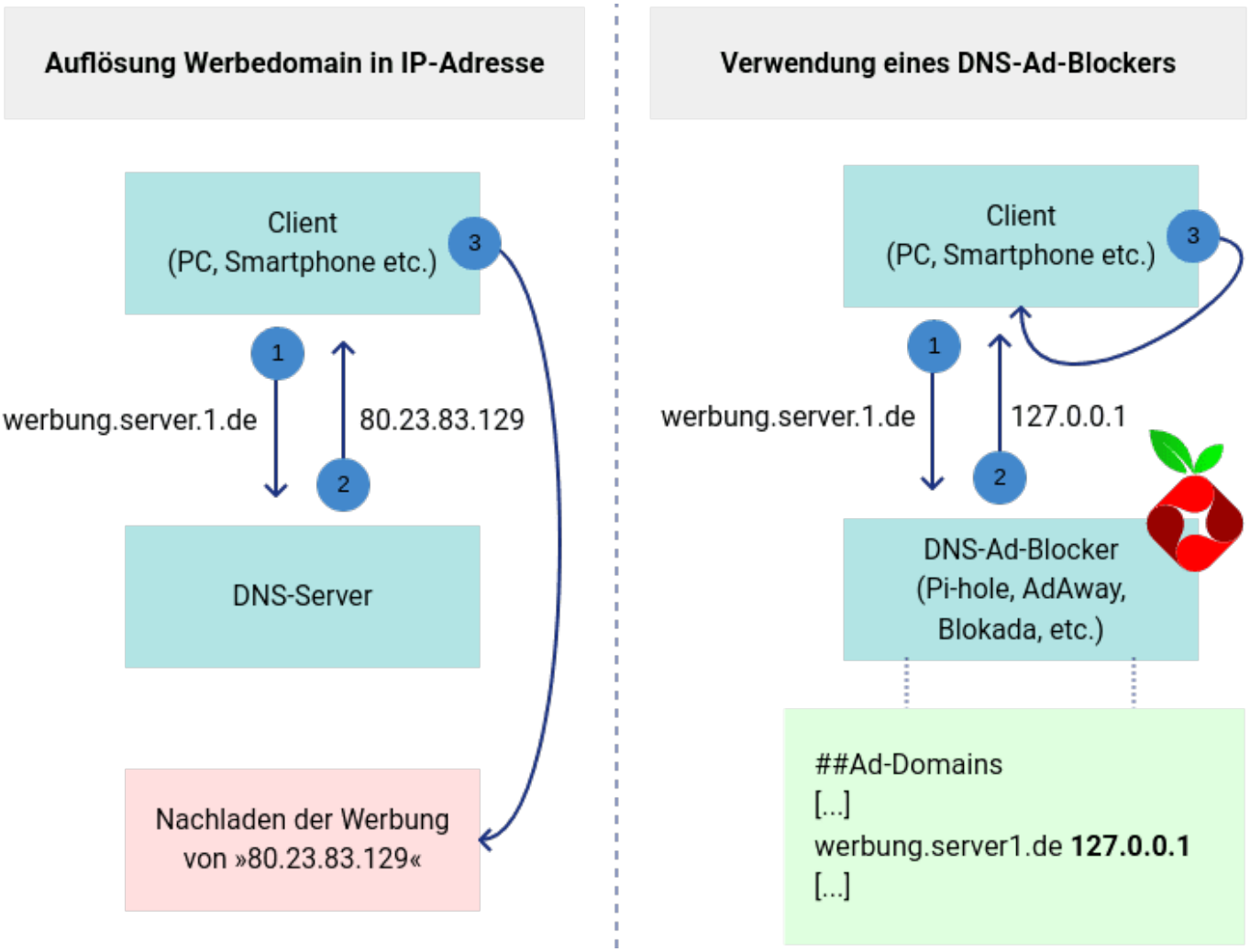
```
werbung.server1.de
```

Dieser Domainname muss jedoch zunächst in eine IP-Adresse übersetzt werden, damit die Werbung von dort aus geladen werden kann. Diese Aufgabe übernimmt das Domain Name System (DNS) – einer der **wichtigsten** Dienste im Internet, der für genau diese Übersetzung zuständig ist. Das Prinzip dahinter kennt jeder: Ihr gebt im Browser eine URI (also den Domainnamen) ein, dieser wird dann von einem DNS-Server in die zugehörige IP-Adresse übersetzt. Namen sind eben leichter zu merken als IP-Adressen. In eurem Router sind daher üblicherweise DNS-Server von eurem Provider hinterlegt oder ihr habt **manuell** eigene eingetragen, die dann anschließend die Adresse »werbung.server1.de« in eine IP-Adresse übersetzen.

Dieses DNS-Prinzip macht sich Pi-hole zunutze. In seinem Speicher verwaltet Pi-hole eine Liste mit Domainnamen, die entweder Werbung ausliefern, den Nutzer tracken/verfolgen oder sich anderweitig negativ auf Sicherheit und Privatsphäre auswirken können. Habt ihr Pi-hole installiert, wird die DNS-Abfrage zunächst mit der intern hinterlegten Liste abgeglichen. Befindet sich die Adresse

```
werbung.server1.de
```

in der Liste bzw. bei einem Treffer, wird die IP-Adresse nicht wie üblich aufgelöst, sondern euer Gerät bzw. die App erhält sinngemäß die Antwort: »Nicht erreichbar« – die Übersetzung in die korrekte IP-Adresse wird also von Pi-hole unterdrückt. Die Folge: Die Werbung kann nicht von der tatsächlichen Quelle bzw. IP-Adresse **nachgeladen** werden. Statt der Werbung sieht der User einen Platzhalter oder schlicht nichts. Ein einfaches Prinzip, das die Werbung schon vor der Auslieferung blockiert – sogar noch vor der Übersetzung in die IP-Adresse:



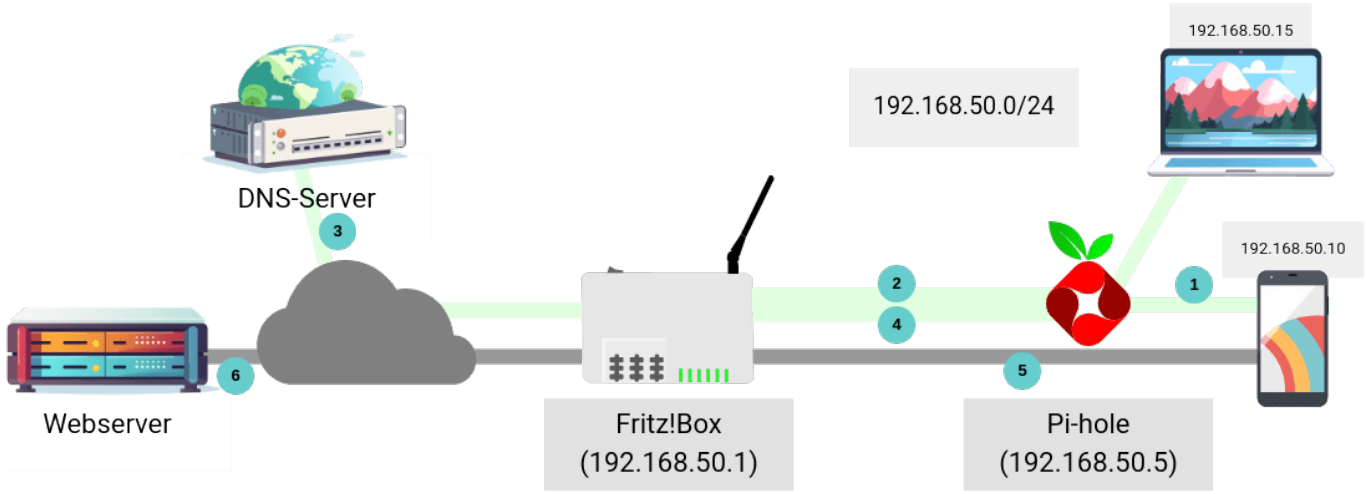
2.2 Netzwerkaufbau

In der Standardkonfiguration stellt Pi-hole seine DNS-Anfragen via UDP/TCP über Port 53 an einen DNS-Server, wie Google, OpenDNS oder Quad9. Die Anfragen erfolgen jedoch unverschlüsselt, also im Klartext. Selbst wenn man einen unzensierten und freien DNS-Server konfiguriert hat, besteht die Möglichkeit, dass jemand die DNS-Anfragen mitliest und auswertet.

Im folgenden Netzwerkaufbau sehen wir, dass der Pi-hole seine DNS-Anfragen nicht direkt an einen DNS-Server eines Anbieters sendet, sondern an die lokale Fritz!Box. Diese wiederum leitet die Anfrage an einen Upstream-DNS-Server weiter. Im Gegensatz zu einem Standard-Pi-hole ist die Fritz!Box nämlich in der Lage, die DNS-Anfragen via DNS over TLS (DoT) zu verschlüsseln. Dies erschwert das Mitlesen bzw. Auswerten des Surfverhaltens – bietet aber natürlich auch keinen hundertprozentigen Schutz (Stichwort: SNI, OCSP). Wer sein Surfverhalten »geheim« halten möchte, muss dann auf den Tor-Browser zurückgreifen und sich an bestimmte Verhaltensregeln halten.

Nachfolgend möchte ich kurz erläutern, was im Netz passiert, wenn ein Smartphone eine Website (www.kuketz-blog.de) aufrufen möchte. Zunächst gibt der Nutzer die Adresse `www.kuketz-blog.de`in die Adresszeile des Browsers ein. Damit der Browser die Seite anzeigen kann, muss die Domain zunächst in die zugehörige IP-Adresse umgewandelt werden. Dazu stellt das Smartphone zunächst eine DNS-Anfrage **[1]** an den Pi-hole. Der Pi-hole prüft nun lokal in seinen Filterlisten, ob es sich bei der Domain um eine Werbe- oder Tracking-Domain handelt. Ist dies nicht der Fall, wird die DNS-Anfrage **[2]** an die Fritz!Box im lokalen Netzwerk weitergeleitet. Die Fritz!Box wiederum sendet die DNS-Anfrage **[3]** nun per DoT verschlüsselt an den hinterlegten (primären) DNS-Server. Nach

Erhalt der zugehörigen IP-Adresse vom DNS-Server, wird die Fritz!Box diese Information nun an den Pi-hole **[4]** weiterreichen und dieser wiederum an das Smartphone. Anschließend kennt der Browser die zugehörige IP-Adresse der Domain `www.kuketz-blog.de` (46.38.242.112) und sendet die HTTP-Anfrage **[5]** zunächst an die lokale Fritz!Box. Diese wiederum leitet die HTTP-Anfrage [6] an den für die IP-Adresse 46.38.242.112 zuständigen Webserver weiter. Anschließend kann die Website an das Smartphone ausgeliefert werden.



Zusammengefasst nehmen DNS-Anfragen den folgenden Weg:

```
Clients -> Pi-hole -> Fritz!Box -> Upstream-DNS-Server
```

Zu beachten gilt, dass sowohl IPv4 als auch IPv6 im Netzwerk benutzt wird. Der Pi-hole bekommt also eine private IPv6-Adresse (Unique Local Address (ULA)) von der Fritz!Box zugewiesen. Geräte im Netzwerk können dann mittels IPv4 oder IPv6 DNS-Anfragen an den Pi-hole senden.

2.3 Minimale Geschwindigkeitseinbußen

Die Fritz!Box ist im dargestellten Netzwerkaufbau eine weitere Zwischenstation, die DNS-Anfragen auf dem Weg zu ihrem Ziel passieren müssen. Das bedeutet zwangsläufig, dass die Pakete etwas länger unterwegs sind. Außerdem wird statt ungesichertem UDP mit TLS gesichertes TCP (DoT) verwendet, was zusätzlichen Overhead bedeutet. Mit dem Benchmark-Tool DNSBench kann man herausfinden, wie groß die zusätzliche Latenz ist – das variiert natürlich je nach verwendetem Upstream-DNS-Server in der Fritz!Box.

Mit den öffentlichen DNS-Servern von AdGuard habe ich Folgendes gemessen:

- **UDP/TCP Port 53 (unverschlüsselt)**: ca. 50 bis 80 Millisekunden
- **DNS over TLS Port 853 (verschlüsselt)**: ca. 90 bis 140 Millisekunden

Die Latenz steigt also tatsächlich – einen spürbaren Unterschied beim Surfen konnte ich allerdings nicht feststellen. Das liegt vermutlich auch daran, dass Pi-hole viele Anfragen ohnehin aus dem Cache liefert.

2.4 Verwendete Software/Version

Nachfolgend möchte ich die verwendete Software kurz vorstellen. Insbesondere die Versionsnummern sind von Interesse, da sich die hier beschriebene Installation durch Updates ändern kann.

- **Basissystem**: Raspberry Pi OS Lite (64 Bit) auf Basis von Debian GNU/Linux Bullseye
- **Pi-hole**: Pi-hole v5.17.1 | FTL v5.23 | Web Interface v5.20.1
- **Fritz!OS**: 7.80
- **Raspberry Pi**: Raspberry Pi 3 Mod. B

Bitte beachten

Die Blog-Beiträge haben nicht wie Enzyklopädie-Einträge (bspw. Wikipedia) den Anspruch, dauerhaft aktuell und richtig zu sein, sondern beziehen sich wie Zeitungsartikel auf den Informationsstand zum Redaktionsschluss. Lediglich die Empfehlungsecke wird regelmäßig aktualisiert.

3. Hardware: Raspberry Pi

Bereits auf einem Raspberry Pi Zero ist Pi-hole lauffähig. Da der Zero allerdings keinen Ethernet-Port hat, würde ich auf ein anderes Modell zurückgreifen. Die aktuellen Raspberry Pi 4 sind für diese Aufgabe fast schon überdimensioniert. Persönlich verwende ich einen Raspberry Pi 3 Mod. B aus dem Jahr 2016. Ihr könnt aber auch schon mit einem Raspberry Pi 1 Mod. B aus dem Jahr 2012 einsteigen – die offizielle Empfehlung lautet nur: Mindestens 512MB RAM.

Hinweis

Auf Wikipedia gibt es eine tabellarische Übersicht, die die Eigenschaften aller Raspberry Pi Modelle übersichtlich darstellt.

Wer noch keinen Raspberry Pi zu Hause hat, greift am besten zu einem KIT. Die offizielle Pi-hole Dokumentation empfiehlt bspw. das Official Pi-hole Raspberry Pi 4 Kit. Daran könnt ihr euch orientieren. Anbei mein Setup:

- Raspberry Pi 3 Modell B (inkl. WLAN)
- Offizielles Micro USB Netzteil 2A – 5V für Raspberry Pi 3
- Offizielles Gehäuse für Raspberry Pi 3 (himbeer / weiß)
- SanDisk Ultra Android microSDHC 16GB bis zu 80 MB/Sek Class 10

Ein Bildschirm oder eine externe Tastatur werden normalerweise nicht benötigt. Nach dem Aufspielen eines Raspberry Pi Images greifen wir über eine Secure Shell (SSH) über das Netzwerk auf den Pi zu.

4. Installation Betriebssystem

Zunächst wird ein Basissystem installiert. Die Raspberry Pi Foundation bietet ein `Raspberry Pi OS Lite (64 Bit)` auf ihrer Website zum Download an. Bei den Lite-Images handelt es sich um ein

minimales System, das aktuell auf Debian Old-Stable (Bullseye) basiert. Der einfachste Weg, das System auf einer SD-Karte zu installieren, ist der Raspberry Pi Imager. Nachfolgend sind die Schritte kurz erklärt.

4.1 Raspberry Pi Imager

- Installiert den Raspberry Pi Imager für euer System (Windows, macOS oder Ubuntu)
- Anschließend startet ihr das Programm und klickt auf `OS WÄHLEN`
- Navigiert zu »Raspberry Pi OS (other) → Raspberry Pi OS Lite (64-bit)« (0.3 GB)
- Danach selektiert ihr im Hauptmenü `SD-KARTE WÄHLEN`. Wählt die korrekte/gewünschte SD-Karte aus, auf dem das System installiert werden soll.
- Vor dem Schreiben des Systems klickt ihr unten rechts auf das Zahnrad und wählt dort Folgendes:
  - SSH aktivieren: **Check**
  - Benutzername und Passwort setzen:
    - Benutzername: **pi**
    - Passwort: **raspberry**
  - Telemetry aktvieren: **Uncheck**
- Klickt auf `SPEICHERN` und anschließend im Hauptmenü auf `SCHREIBEN`. Nachdem der Vorgang abgeschlossen ist, legt ihr die SD-Karte in den Raspberry Pi ein, verbindet das Ethernetkabel mit der Fritz!Box und schließt das Netzteil an.

Hinweis

Es gibt auch ein Video, das die Schritte nochmal visuell darstellt.

4.2 Alternativ: Manuelle Installation des Images

Ihr könnt die Schritte natürlich auch manuell ausführen. Das sieht dann wie folgt aus:

- Download Raspberry Pi OS Lite (64-bit)
- Zip-File entpacken
- Schreibt das Image auf die SD-Karte. Unter Linux (Terminal) funktioniert das bspw. wie folgt:

```
sudo dd bs=4M if=/Pfad/zum/Image/2023-05-03-raspios-bullseye-arm64-lite.img
of=/dev/sdX conv=fsync
```

Stellt unbedingt sicher, dass ihr bei `/dev/sdX` das korrekte Device/Gerät angeben habt (also die SD-Karte).

- Nachdem das Image auf die SD-Karte geschrieben wurde, könnt ihr auf das Dateisystem zugreifen. Navigiert auf die Partition **/boot[fs]** und legt dort folgende Dateien an:
  - **ssh**: Legt eine Datei mit der Bezeichnung `ssh` an. Dies ist erforderlich, um anschließend mittels Secure Shell (SSH) auf den Raspberry Pi zugreifen zu können. Unter Windows gibt es diesen Ordner nicht, hier legt ihr die Datei direkt im Hauptverzeichnis der SD-Karte an.
  - **userconf.txt**: Anschließend wird eine Datei mit der Bezeichnung `userconf.txt` angelegt. Dort tragt ihr Folgendes ein:

```
pi:$6$T6s1r19kJcX4WPpC$W5K/SPq6HE84yK.GwLuS2tqQhV6W4g6vHe/bIsBeE2rHJpbRXeQIE
AjP0KnlCU5ifFfq.t96.KWX81iGO3kQV/
```

Der Nutzername ist dann `pi` und das initiale Passwort `raspberry`.

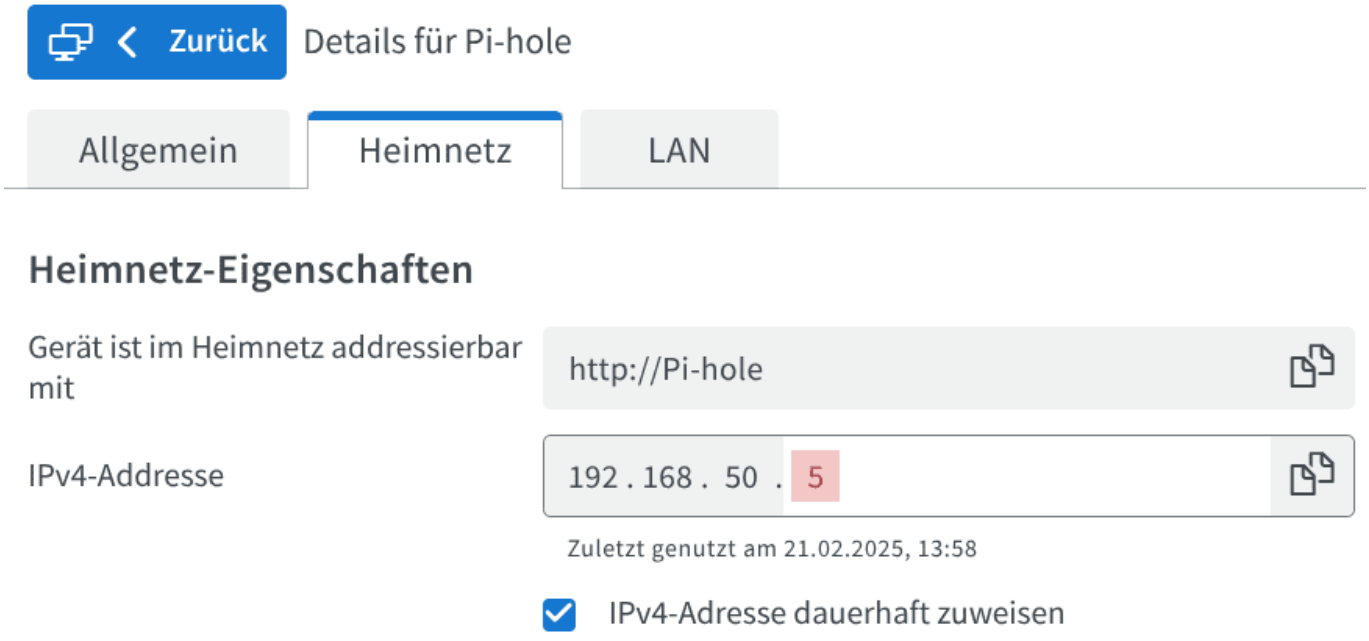
- Nachdem die Dateien angelegt sind, legt ihr die SD-Karte in den Raspberry Pi ein, verbindet das Ethernetkabel mit der Fritz!Box und schließt das Netzteil an.

4.3 Netzwerkverbindung Raspberry Pi

Ihr könnt den Raspberry Pi nun zum ersten Mal starten. Im Normalfall sollte eure Fritz!Box einen bestimmten IP-Adressbereich reserviert haben, um mittels DHCP eine IP-Adresse an den Pi zu vergeben. Über das Webinterface der Fritz!Box vergeben wir nachfolgend eine feste IP-Adresse an den Pi:

- Meldet euch am Webinterface der Fritz!Box an und navigiert zu `Heimnetz → Netzwerk`. Dort sollte der Pi in der Liste der aktiven Verbindungen erscheinen. Über das Bleistiftsymbol am Ende der Zeile gelangt man zu den Details des Gerätes. Legt dort folgende Optionen/Einstellungen fest:
  - Setzt ein Häkchen bei Internetnutzung priorisiert
  - Verändert unter IPv4-Adresse die letzte Zahl – nach dem dritten Punkt – und stellt damit eine feste IP-Adresse ein
  - Klickt anschließend auf Übernehmen

Das sieht dann unter FRITZ!OS 7.57 wie folgt aus:



Hinweis

Die hier verwendete IP-Adresse (`192.168.50.5`) müsst ihr in der Folge dann immer entsprechend durch eure selbst vergebene ersetzen.

Damit die Fritz!Box dem Raspberry Pi die eingestellte IP-Adresse zuweist, müsst ihr das Netzwerkkabel ziehen oder den Pi kurz vom Strom trennen bzw. neu starten. Danach ist euer Raspberry Pi unter der vergebenen IP-Adresse per SSH erreichbar.

4.4 Einrichtung Raspberry Pi

Unter macOS oder Linux starten wir eine SSH-Sitzung über das Terminal – für Windows kann PuTTY verwendet werden. Die Zugangsdaten lauten: Benutzername (`pi`) / Passwort (`raspberry`)

```
ssh pi@192.168.50.5
```

Zunächst ändern wir das initial vergebene Passwort:

```
passwd
```

Die Zeitzone (Europe/Berlin) passen wir mit raspi-config an:

```
sudo raspi-config
Localisation Options -> Timezone -> Europe -> Berlin
Finish
```

Anschließend aktualisieren wir die Paketdatenbank und bringen die vorinstallierte Software auf den neuesten Stand:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get dist-upgrade
sudo apt-get autoremove
```

Das dauert für gewöhnlich etwas. Nach Abschluss des Vorgangs wird ein Neustart eingeleitet:

```
sudo reboot
```

5. Pi-hole

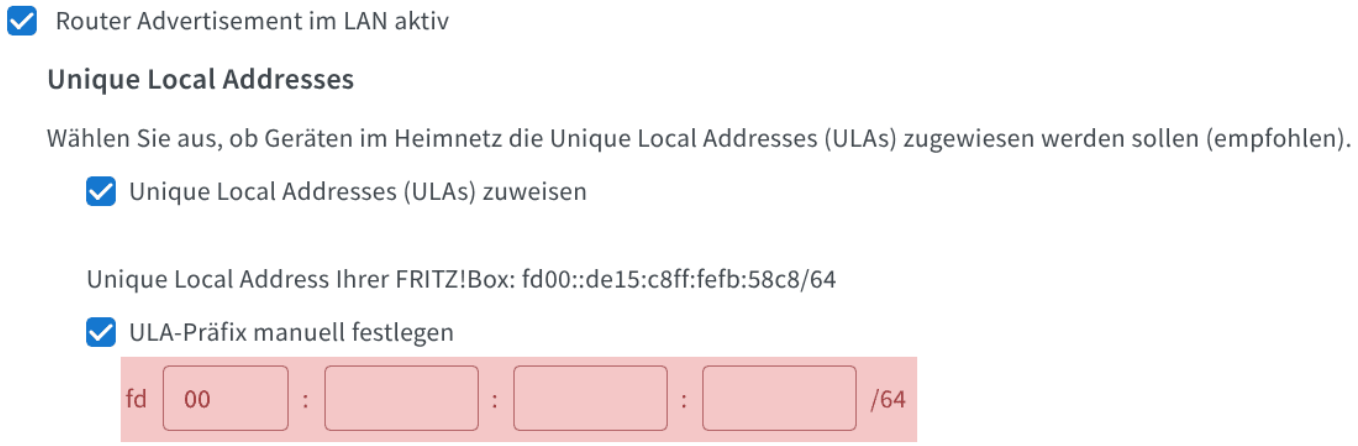
Vor der Installation von Pi-hole konfigurieren wir die Fritz!Box so, dass der Raspberry Pi bereits eine private IPv6-Adresse (ULA) zugewiesen bekommt. Über diese IPv6-Adresse ist der Pi-hole anschließend via IPv6 für die Clients im Netzwerk erreichbar.

5.1 Vor-Einstellungen Fritz!Box

Navigiert zu `Heimnetz → Netzwerk → Netzwerkeinstellungen → IPv6-Einstellungen` (fast ganz unten). Passt die Einstellungen dort nun wie folgt an:

- Router Advertisement im LAN aktiv: **Check**
  - Unique Local Addresses (ULA) immer zuweisen: **Check**
  - ULA-Präfix manuell festlegen
    - **fd 00 : [leer] : [leer] : [leer] /64**

Das sieht dann wie folgt aus:



5.2 Installation

Wir melden uns erneut über SSH am Pi-hole an und starten die Installation mit folgendem Befehl:

```
sudo curl -sSL https://install.pi-hole.net | bash
```

Der Installationsassistent bietet Optionen zur Konfiguration an. Diese werden wie folgt vorgenommen:

```
Static IP Address**
   Yes**: Set static IP using current values
Upstream DNS Provider
   **OpenDNS (ECS, DNSSEC)
**Blocklists
   **Yes (include StevenBlack's)**
Admin Web Interface
   **Yes**
Web Server
   **Yes**
Enable Logging
   **Yes**
Privacy mode FTL
   **Show everything (private Installation)**
```

Nach der Installation werden die IPv4- und die IPv6-Adresse des Pi-Holes angezeigt. Notiert euch beide Adressen oder macht einen Screenshot, da beide Angaben noch benötigt werden. Ebenso die Adresse, unter der das Webinterface erreichbar ist. Das Passwort für das Webinterface ist standardmäßig etwas kurz, wir können dies mit folgendem Befehl ändern und ein sicheres Passwort vergeben:

```
pihole -a -p
```

Wer die Informationen mit den IP-Adressen zu schnell wegklickt, sollte nicht verzagen: Die IP4- und IPv6-Adressen des Pi-hole können ebenfalls über das Terminal ermittelt werden.

- **IPv4**:

```
ip address | grep "inet 192"
```

- **IPv6**:

```
ip address | grep "inet6 fd"
```

6. Konfiguration

Wir wollen nun erreichen, dass alle DNS-Anfragen von den Clients zunächst an den Pi-hole gestellt werden. Dieser leitet die Anfrage dann an die Fritz!Box weiter, die gegenüber dem Pi-hole als Upstream-DNS-Server fungiert. In der Fritz!Box wird der Pi-hole per DHCP als lokaler DNS-Server für alle Clients bekannt gemacht. Falls kein DHCP verwendet wird, muss der DNS-Server auf den Clients manuell geändert werden.

6.1 Fritz!Box: Pi-hole als DNS-Server

Nachfolgend wird Pi-hole als lokaler DNS-Server für alle Clients bekannt gemacht.

Zunächst werden die IPv4-Netzwerkeinstellungen auf der Fritz!Box angepasst. Navigiert dazu zu `Heimnetz → Netzwerk → Netzwerkeinstellungen → IPv4-Einstellungen` (fast ganz unten):

- Lokaler DNS-Server: **192.168.50.5** [hier die IPv4-Adresse des Pi-hole eintragen]

Anschließend passen wir die IPv6-Netzwerkeinstellungen über `Heimnetz → Netzwerk → Netzwerkeinstellungen → IPv6-Einstellungen` (fast ganz unten) wie folgt an:

- DNSv6-Server auch über Router Advertisement bekanntgeben (RFC 5006): **Check**
- Lokaler DNSv6-Server: **fd00 : 0 : 0 : 0 : eae7 : a43a : 2f8c : 9fd6** [hier die IPv6-Adresse des Pi-hole eintragen]

6.2 Fritz!Box: DoT-fähige DNS-Server hinterlegen

DNS-Anfragen aus dem lokalen Netzwerk sollen DoT-verschlüsselt an einen Upstream-DNS-Server gesendet werden. Dies übernimmt die Fritz!Box. Dazu öffnet man die Einstellungen unter `Internet → Zugangsart → DNS-Server`. Nachfolgend habe ich die DNS-Server von AdGuard (bevorzugt) und dnsforge.de (alternativ) verwendet.

- DNSv4-Server
  - Andere DNSv4-Server verwenden: **Check**

        - Bevorzugter DNSv4-Server: **94.140.14.140**
        - Alternativer DNSv4-Server: **176.9.93.198**
- DNSv6-Server
    - Andere DNSv6-Server verwenden: **Check**
        - Bevorzugter DNSv6-Server: **2a10:50c0::1:ff**
        - Alternativer DNSv6-Server: **2a01:4f8:151:34aa::198**
- DNS over TLS (DoT)
    - Verschlüsselte Namensauflösung im Internet (DNS over TLS): **Check**
        - Zertifikatsprüfung für verschlüsselte Namensauflösung im Internet erzwingen: **Check**
    - Auflösungsnamen der DNS-Server

```
unfiltered.adguard-dns.com
dnsforge.de
```

6.3 Pi-hole

Abschließend ist noch eine Anpassung über das Webinterface des Pi-hole erforderlich. Während der Installation wurde als Upstream-DNS-Server `OpenDNS (ECS, DNSSEC)` gewählt, dies muss angepasst werden. Wir geben folgende Adresse in den Browser ein und melden uns an:

```
192.168.50.5/admin
```

Unter `Settings → DNS` nehmen wir folgende Einstellungen vor:

- Alle Häkchen bei OpenDNS (ECS, DNSSEC) und auch bei allen weiteren DNS-Servern (falls vorhanden) entfernen
- Custom 1 (IPv4): **192.168.50.1** (IPv4-Adresse der Fritz!Box)
- Custom 3 (IPv6): **fd00::de15:c8ff:fe13:9af5** (IPv6-Adresse der Fritz!Box)

Hinweis

Die Unique Local Address bzw. die private IPv6-Adresse der Fritz!Box könnt ihr wie folgt herausfinden. Meldet euch im Webinterface der Fritz!Box an und öffnet die Einstellungen unter `Netzwerk → Netzwerkeinstellungen → IPv6-Einstellungen`. Direkt oben unter »Unique Local Addresses« findet ihr hinter der Bezeichnung `Unique Local Address Ihrer FRITZ!Box` die IPv6-Adresse eurer Fritz!Box.

Upstream DNS Servers

IPv4		IPv6		Name
☐	☐	☐	☐	Google (ECS, DNSSEC)
☐	☐	☐	☐	OpenDNS (ECS, DNSSEC)
☐	☐			Level3
☐	☐			Comodo
☐	☐	☐	☐	DNS.WATCH (DNSSEC)
☐	☐	☐	☐	Quad9 (filtered, DNSSEC)
☐	☐	☐	☐	Quad9 (unfiltered, no DNSSEC)
☐	☐	☐	☐	Quad9 (filtered, ECS, DNSSEC)
☐	☐	☐	☐	Cloudflare (DNSSEC)

ECS (Extended Client Subnet) defines a mechanism for recursive resolvers to send partial client IP address information to authoritative DNS name servers. Content Delivery Networks (CDNs) and latency-sensitive services use this to give geo-located responses when responding to name lookups coming through public DNS resolvers. *Note that ECS may result in reduced privacy.*

Custom DNS servers (1 custom server enabled) −

The following list contains all DNS servers selected above. Furthermore, you can add your own custom DNS servers here. The expected format is one server per line in form of `IP#port`, where the `port` is optional. If given, it has to be separated by a hash `#` from the address (e.g. `127.0.0.1#5335` for a local `unbound` instance running on port `5335`). The port defaults to 53 if omitted.

```
192.168.50.1
fd00::de15:c8ff:fe13:9af5
```

Anschließend wird ganz unten auf der Seite `Use Conditional Forwarding` angekreuzt und die folgenden Werte eingetragen:

- Local network in CIDR notation: **192.168.50.0/24** (an euer Netzwerk anpassen)
- IP address of your DHCP server (router): **192.168.50.1** (IPv4-Adresse eurer Fritz!Box)
- Local domain name (optional): **fritz.box**

Nach einem Klick auf `Save` ist euer Pi-hole bereits einsatzbereit. Startet eure Geräte im Netzwerk neu, damit ihnen der neue DNS-Server (Pi-hole) per DHCP mitgeteilt wird. Danach werden ausgehende DNS-Anfragen über das Pi-hole gefiltert.

7. Anpassungen: Filterlisten, Updates und Co. [optional]

7.1 Filterlisten

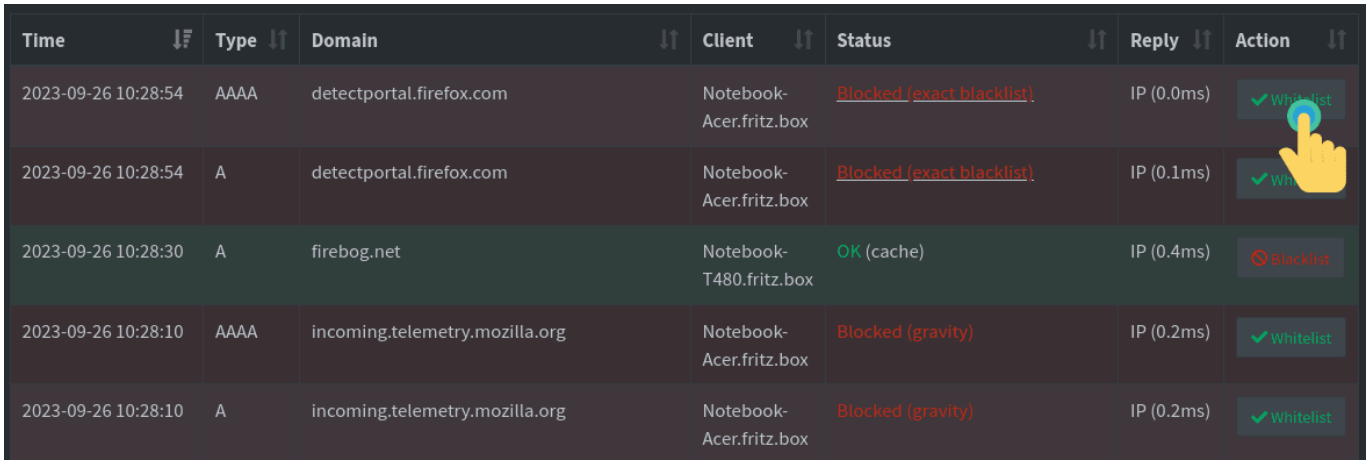
Die mitgelieferte Filterliste (StevenBlack) entfernt bereits eine Menge Werbung, Tracker und anderen unnötigen Netzballast. Im Webinterface von Pi-hole kann man über das Menü `Adlists` weitere Filterlisten hinzufügen. Persönlich verwende diese drei Filterlisten:

- StevenBlack: Unified hosts + fakenews + gambling + porn + social
- hagezi -> Pro++
- hagezi -> Amazon (Devices, Shopping, Video)

Natürlich können auch andere/weitere (kombinierte) Filterlisten (bspw. The Firebog, WindowsSpyBlocker (Hosts)) aktiviert bzw. hinzugefügt werden. Eventuelle Überschneidungen werden von Pi-hole automatisch entfernt – doppelte Einträge wären für die Verarbeitung der Filterlisten zu ineffizient. Nach dem Hinzufügen der Filterlisten kann ein manuelles Update angestoßen werden, damit diese sofort verwendet werden. Öffnet dazu den Menüpunkt `Tools → Update Gravity` und klickt auf den `Update-Button`. Standardmäßig aktualisiert Pi-hole die Filterlisten einmal pro Woche.

Durch die Aktivierung von Filterlisten kann es zum sogenannten »Overblocking« kommen. Das bedeutet, dass Domains, die für die Funktionalität einer App/Website notwendig sind, fälschlicherweise gefiltert werden. Es muss dann im Einzelfall entschieden werden, welche Domain(s) über das Webinterface in Pi-hole freigegeben werden. Meldet euch dazu am Pi-hole-Webinterface an und klickt im Menü auf `Query Log` – dort werden alle DNS-Anfragen protokolliert. Am Ende jeder Zeile kann eine Aktion für die Domain ausgewählt werden:

- **Blacklist**: Die Domain wird in Zukunft gefiltert, d.h. blockiert
- **Whitelist**: Die Domain wird in Zukunft nicht mehr gefiltert, d.h. nicht mehr blockiert



7.2 Update-Verhalten anpassen [Fortgeschrittene]

Über die Konfigurationsdatei `/etc/cron.d/pihole` kann das Update-Verhalten von Pi-hole an die eigenen Bedürfnisse angepasst werden. Persönlich habe ich die Aktualisierung von Filterlisten bspw. auf Sonntagabend um 22:30 Uhr gelegt:

```
sudo nano /etc/cron.d/pihole
```

```
# Pi-hole update ad sources
30 22 * * 0   root    PATH="$PATH:/usr/sbin:/usr/local/bin/" pihole
updateGravity>/var/log/pihole/pihole_updateGravity.log || cat
/var/log/pihole/pihole_updateGravity.log
```

7.3 Energie/Strom sparen [Fortgeschrittene]

Ein Raspberry Pi verfügt über verschiedene Schnittstellen wie Audio, Bluetooth, WiFi etc. Die meisten Schnittstellen sind für den Betrieb des Pi-hole nicht notwendig. Das Deaktivieren dieser Schnittstellen reduziert den Stromverbrauch des kleinen Einplatinencomputers. Nach dem Abschalten von Audio, Bluetooth, WiFi und HDMI habe ich im Betrieb noch einen Verbrauch von ca. 1,2 bis 1,5 Watt gemessen – je nach Auslastung. Das bedeutet, dass der Stromverbrauch eines Raspberry Pi 3 Mod. B von ca. 1,8 Watt um ca. 0,5 bis 0,6 Watt reduziert werden kann. Die notwendigen Anpassungen können direkt in einer Konfigurationsdatei (config.txt) vorgenommen werden, nachdem man sich per SSH mit dem Pi verbunden hat:

```
sudo nano /boot/config.txt
```

```
# Disable analog audio
dtparam=audio=off
# Disable audio via HDMI
dtoverlay=vc4-kms-v3d,noaudio
# Disable Bluetooth, WiFi and HDMI
dtoverlay=disable-bt
dtoverlay=disable-wifi
hdmi_blanking=1
```

Die Zeilen mit `dtparam=audio` und `dtoverlay` sind bereits vorhanden und müssen angepasst werden. Die Befehle zum Deaktivieren von Bluetooth, WiFi und HDMI können einfach am Ende der Konfigurationsdatei hinzugefügt werden.

7.4 Schreibzugriffe auf SD-Karte minimieren [Fortgeschrittene]

Permanente Schreibvorgänge können die Lebensdauer einer SD-Karte verkürzen. Nachfolgend werden zwei Möglichkeiten vorgestellt, die Lebensdauer der SD-Karte in Kombination mit Pi-hole (möglicherweise) zu verlängern.

Standardmäßig schreibt Pi-hole jede Minute in eine Log-Datei bzw. eine SQL-Lite-Datenbank (SQLite3). Durch die zeitliche Streckung dieses Schreibvorgangs auf 30 Minuten kann die SD-Karte etwas geschont werden. Außerdem werden wir die Langzeitdaten (Long-term Data), die Pi-hole sammelt, von 365 Tagen auf 30 Tage reduzieren:

```
sudo nano /etc/pihole/pihole-FTL.conf
```

```
#; How often do we store queries in FTL's database [minutes] | Default: 1.0
DBINTERVAL=30
#; IP addresses older than the specified number of days are removed from
database | Default: 365
MAXDBDAYS=30
```

Zusätzlich besteht die Möglichkeit, bestimmte Ordner, auf die häufig Schreibzugriffe erfolgen, in eine RAM-Disk auszulagern. Mit dem Linux-Tool tmpfs ist dies einfach zu bewerkstelligen. Auf Linux-Systemen, die journald für die Protokollierung verwenden, hat dieser Schritt jedoch wenig Einfluss auf eine mögliche Verlängerung der Lebensdauer. Die Schreibvorgänge sind in der Regel gebündelt und erfolgen ca. alle 5 Minuten. Wer dies dennoch umsetzen möchte, kann bspw. die Verzeichnisse `/tmp` und `/var/log` in eine tmpfs-RAM-Disk mit jeweils 100 MB RAM auslagern:

```
sudo nano /etc/fstab
```

```
tmpfs   /tmp    tmpfs   defaults,noatime,nosuid,size=100m   0   0
tmpfs   /var/log   tmpfs   defaults,noatime,nosuid,mode=0755,size=100m   0
0
```

Hinweis

Nach einem Absturz oder Neustart des Gerätes sind Dateien/Informationen in diesen Verzeichnissen verschwunden. Entweder man erstellt sich einen Cronjob, der die Daten regelmäßig vom RAM auf die SD-Karte schreibt oder man verwendet ein Tool wie Log2Ram.

Alle Lese- und Schreibvorgänge auf die SD-Karte (bzw. Fesplatte) können übrigens mit dem Tool fatrace dargestellt werden:

```
sudo apt-get install fatrace
```

Mit dem folgenden Kommando kann fatrace so eingestellt werden, dass nur Schreiboperationen ausgegeben werden:

```
sudo fatrace --filter=W --timestamp
```

Das sieht dann bspw. wie folgt aus:

> 09:14:49.934532 pihole-FTL(572): CW /etc/pihole/pihole-FTL.db
> 09:14:50.946265 php-cgi(654): CW /etc/pihole/gravity.db
> […]

7.5 Automatische Updates: System und Pi-hole [Fortgeschrittene]

Beim Thema Updates bzw. Aktualisierungen müssen wir zunächst zwischen verschiedenen Komponenten unterscheiden:

- Updates des Basissystems bzw. der installierten Debian-Pakete
- Update des Pi-hole bzw. FTL
- Update der hinterlegten Filterlisten

Die Filterlisten werden von Pi-hole standardmäßig einmal pro Woche aktualisiert. Diesen Punkt können wir also abhaken. Zur automatischen Aktualisierung von Pi-hole bzw. seiner Komponenten wie FTL sagt das Pi-hole-Projekt:

> This is technically possible, but we do not recommend auto-updating Pi-hole. Some updates are breaking, and if you wake up in the morning with an update that doesn't meet your needs (and no backup) you will have a problem. The dashboard will notify you when an update is available. Read the release notes, then if you want to update do so manually.

Über die Weboberfläche von Pi-hole wird man über ein anstehendes Update demnach informiert. Nach Durchsicht der Versionshinweise bzw. der Änderungen sollte das Update dann vom Benutzer manuell angestoßen werden.

Das Basissystem hingegen, das auf Debian basiert, kann nach meiner Erfahrung automatisch (Unattended Upgrades) aktualisiert werden. Dazu sind folgende Schritte notwendig:

Zunächst werden die Pakete `unattended-upgrades` und `apt-listchanges` installiert:

```
sudo apt-get install unattended-upgrades apt-listchanges
```

Danach passen wir die Konfigurationsdatei an:

```
sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

Damit automatisch Pakete vom Raspbian-Projekt eingespielt werden, ist die folgende Anpassung notwendig. Nach

```
"origin=Debian,codename=${distro_codename}-security,label=Debian-Security";
```

werden die folgenden zwei Zeilen hinzugefügt:

```
"origin=Raspbian,codename=${distro_codename},label=Raspbian";
"origin=Raspberry Pi Foundation,codename=${distro_codename},label=Raspberry
Pi Foundation";
```

Mit dem nachfolgenden Befehl kann eine Paketprüfung manuell angestoßen werden:

```
sudo unattended-upgrade -d
```

Hinweis

UnattendedUpgrades aktualisiert nur die Pakete der aktuell installierten Debian/Rasbian-Version. Upgrades auf neue Debian-Versionen (bspw. von Bullseye auf Bookworm) müssen manuell durchgeführt werden.

8. Benötige ich noch einen weiteren AdBlocker?

Pi-hole blockiert Werbung und Tracker auf DNS-Ebene. Insbesondere Geräte, auf denen sich bspw. kein AdBlocker installieren lässt, profitieren also vom Pi-hole. Auf Computern oder Geräten, auf denen AdBlocker wie uBlock Origin im Browser installiert werden können, sollte dies dennoch weiterhin geschehen. Denn diese verfügen über zusätzliche Filtermechanismen und Logik. Im Idealfall blockieren die Browser-Add-ons also all jene Gemeinheiten, die der Pi-hole auf DNS-Ebene nicht erkennt. Der Pi-hole ist eure erste Verteidigungslinie – ein Browser-Add-on wie uBlock Origin »eliminiert« dann den Rest.

9. Fazit

In diesem Artikel wurde gezeigt, wie man ein Pi-hole in das (Heim-)Netzwerk einer Fritz!Box integriert. Alle DNS-Anfragen werden TLS-verschlüsselt (DoT) an Upstream-DNS-Server gesendet, die in der Fritz!Box hinterlegt sind. Somit agiert Pi-hole nur im lokalen Netzwerk und filtert zuverlässig Werbung/Tracker heraus.

Im nächsten Teil der Artikelserie wird Pi-hole in Kombination mit unbound konfiguriert, um die Abhängigkeit zu den DNS-Servern bzw. Providern aufzulösen. Alle DNS-Anfragen werden dann direkt an die Root-Server des Internets gestellt. Einsteigern empfehle ich jedoch, das vorliegende Setup umzusetzen.

= Pi-hole: Einrichtung und Konfiguration mit unbound – AdBlocker Teil2

1. unbound

Im ersten Teil der Artikelserie »AdBlocker« wurde gezeigt, wie man ein Pi-hole in das (Heim-)Netzwerk einer Fritz!Box integriert. Nach der Umsetzung werden Werbung, Tracker, Affiliates, Telemetrie, Malware, Scam etc. zuverlässig herausgefiltert – das Internet wird für Clients im Netzwerk ein Stück sauberer.

Der vorliegende Artikel beschreibt, wie man den rekursiven DNS-Server unbound auf einem Pi-hole konfiguriert. Dadurch wird nicht nur die Abhängigkeit von öffentlichen DNS-Servern aufgehoben, sondern auch die DNS-Abfragen werden schneller beantwortet. Die Konfiguration richtet sich an fortgeschrittene Benutzer – Anfängern empfehle ich, das Setup aus dem ersten Teil zu verwenden.

2. Was wir erreichen wollen

Voraussetzung für die Installation bzw. Konfiguration von unbound ist ein bereits funktionierender Pi-hole, wie er im ersten Teil vorgestellt wurde. Habt ihr dieses Setup umgesetzt, könnt ihr nahtlos mit den folgenden Schritten fortgefahren. Vor der Umsetzung empfehle ich, den ersten Teil noch einmal in Ruhe zu lesen.

Unbound ändert nichts am ursprünglichen Ziel: ausgehende DNS-Anfragen an Werbe- und Tracking-Domains zu blockieren. Wir verzichten jedoch auf den Mittelsmann – den vorgeschalteten DNS-Server – und beseitigen damit die Abhängigkeit von einem Anbieter. Durch die Installation von unbound werden wir sozusagen zu unserem eigenen DNS-Server-Anbieter.

2.1 Vor- und Nachteile von unbound

Unbound ist ein validierender (DNSSEC), rekursiver DNS-Server, der Anfragen in einem Cache zwischenspeichert. Entwickelt wird die Software von NLnet Labs – der Quellcode ist vollständig auf GitHub verfügbar und steht unter der BSD-Lizenz. Unbound läuft auf allen Linux- und BSD-Distributionen sowie auf macOS. Es ist in allen Standard-Repositories der gängigen Linux-Distributionen enthalten.

Was ändert sich aber durch den Einsatz von unbound auf einem Pi-hole? Diese Frage wird im Folgenden näher beleuchtet. Starten wir mit einem Zitat aus der Pi-hole Dokumentation:

> Pi-hole enthält einen DNS-Server zur Zwischenspeicherung und Weiterleitung, der als FTLDNS bekannt ist. Nach Anwendung der Sperrlisten leitet er die von den Clients gestellten Anfragen an konfigurierte Upstream-DNS-Server weiter. Wie jedoch von mehreren Benutzern in der Vergangenheit erwähnt wurde, führt dies zu einigen Bedenken hinsichtlich des Datenschutzes, da sich letztlich die Frage stellt: Wem kann man vertrauen? In letzter Zeit sind immer mehr kleine (und nicht so kleine) DNS-Upstream-Anbieter auf dem Markt aufgetaucht, die mit kostenlosen und privaten DNS-Diensten werben, aber woher weiß man, dass sie ihre Versprechen halten? Richtig, man kann es nicht.

Lasst uns das Problem klar benennen: Es geht um Vertrauen. Wir müssen darauf vertrauen, dass unser DNS-Server-Anbieter unsere DNS-Anfragen nicht aufzeichnet und/oder analysiert. Persönlich halte ich die DNS-Server-Anbieter in der Empfehlungsecke für vertrauenswürdig, möchte aber betonen, dass ich nicht mit absoluter Sicherheit sagen kann, dass keiner dieser Anbieter jemals sein Versprechen bricht und beispielsweise DNS-Anfragen protokolliert.

Mit unbound lösen wir die Abhängigkeit zu einem DNS-Server-Anbieter auf. Daraus ergeben sich folgende Vorteile:

- **Privatsphäre**: Unbound wird die Namensauflösung über einen Root-Nameserver starten und sich bis zum zuständigen autoritativen Nameserver »durchfragen«, der den Domainnamen in die zugehörige IP-Adresse übersetzen kann. Wir verzichten also auf einen zentralen DNS-Mittelsmann, der unsere DNS-Anfragen aufzeichnen und/oder analysieren könnte.
- **Geschwindigkeit**: Pi-hole selbst hat bereits einen Cache für DNS-Anfragen integriert. Mit unbound und den entsprechenden Konfigurationsparametern (bswp. aggressive-nsec, prefetch, serve-expired etc.) lassen sich die Antwortzeiten abermals minimieren.

Der Verzicht auf einen DNS-Mittelsmann geht allerdings auch mit folgendem Nachteil einher:

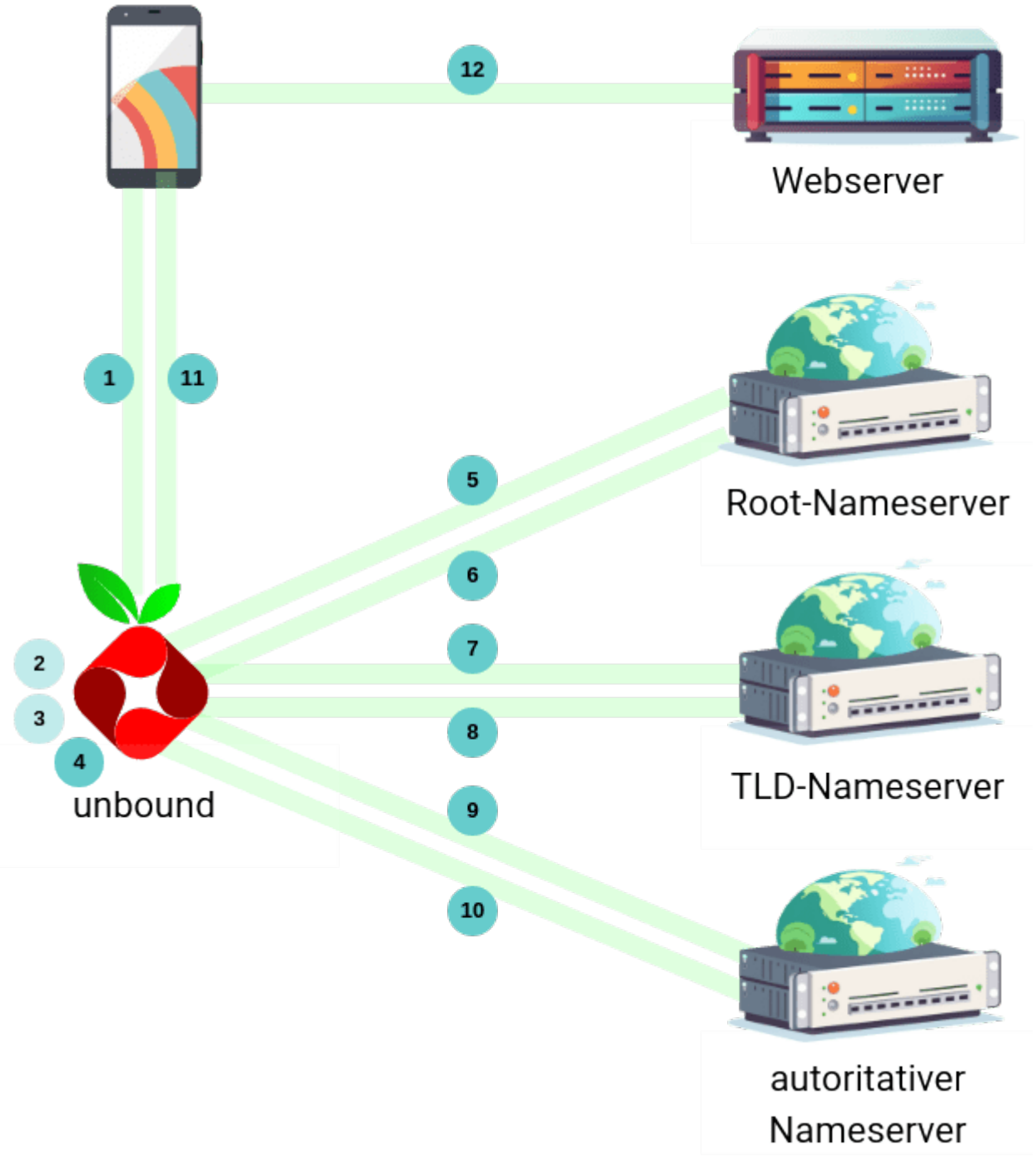
- **Keine Verschlüsselung**: Anfragen an die Root-Nameserver bzw. autoritativen Nameserver erfolgen unverschlüsselt via UDP/TCP über Port 53. Es besteht also die Möglichkeit, dass jemand die DNS-Anfragen mitliest und auswertet. Um dieses Risiko zu verringern, wird im ersten Teil ein Setup beschrieben, bei dem die DNS-Anfragen via DNS over TLS (DoT) verschlüsselt werden. Diese Möglichkeit können wir im folgenden Setup leider nicht anwenden. Unbound unterstützt zwar DNS over TLS (DoT) und auch DNS over HTTPS (DoH), aber die Root-Nameserver unterstützen diese Protokolle (noch) nicht. Wenn wir also auf einen DNS-Mittelsmann verzichten wollen, müssen wir auch auf die Verschlüsselung unserer DNS-Anfragen verzichten. Der Einsatz von DNSSEC unter unbound stellt aber zumindest sicher, dass die zurückgelieferten Daten echt (Authentizität) und unverändert (Integrität) sind.

Hinweis

In diesem Zusammenhang möchte ich noch einmal an meinen Hinweis im ersten Teil erinnern. Wer wirklich Gefahr läuft, dass sein Surfverhalten mitgelesen und ausgewertet wird, sollte auf den Tor-Browser zurückgreifen und bestimmte Verhaltensregeln beachten.

2.2 Ablauf einer DNS-Abfrage unbound | Pi-hole

Nachfolgend ist der Ablauf einer DNS-Abfrage von unbound in Kombination mit Pi-hole für die Domain `kuketz-blog.de` dargestellt:



- **[1]** Ein Client fragt den Pi-hole nach der Domain `kuketz-blog.de`
- **[2]** Pi-hole überprüft seinen Cache und antwortet, wenn die zugehörige IP-Adresse bereits bekannt ist
- **[3]** Pi-hole überprüft seine Filterliste und antwortet, wenn die Domain gesperrt ist
- **[4]** Da keines der beiden Szenarien zutrifft, sendet Pi-hole die Anfrage an den (lokalen) rekursiven DNS-Server – also an unbound
- **[5]** unbound sendet die Anfrage nun an einen Root-Nameserver und fragt, wer für die Top-Level-Domain (TLD) `.de` zuständig ist
- **[6]** Der Root-Nameserver antwortet mit einem Verweis auf einen TLD-Server für `.de`

- **[7]** An den TLD-Server sendet unbound anschließend eine Anfrage, wer die Domain `kuketz-blog.de` verwaltet
- **[8]** Der TLD-Server wiederum antwortet mit einem Verweis auf den autoritativen Nameserver für `kuketz-blog.de`
- **[9]** Im letzten Schritt fragt unbound nun den autoritativen Nameserver nach der IP-Adresse für `kuketz-blog.de`
- **[10]** Der autoritative Nameserver übermittelt die IP-Adresse der Domain `kuketz-blog.de` an unbound
- **[11]** unbound gibt die IP-Adresse an Pi-hole weiter, der wiederum dem Client die Antwort auf seine Anfrage mitteilt
- **[12]** Der Client bzw. Browser kann nun über die IP-Adresse die Website `kuketz-blog.de` laden/darstellen

In diesem Ablauf werden drei verschiedene Arten von DNS-Servern benannt: Root-Nameserver, TLD-Nameserver und autoritative Nameserver. Um die Unterschiede zwischen diesen drei Typen besser zu verstehen, lest bitte den Artikel »Welche verschiedenen Typen von DNS-Servern gibt es?«.

Hinweis

Der Ablauf beinhaltet die QNAME Minimisation.

2.3 Verwendete Software/Version

Nachfolgend möchte ich die verwendete Software kurz vorstellen. Insbesondere die Versionsnummern sind von Interesse, da sich die hier beschriebene Installation durch Updates ändern kann.

- **Basissystem**: Raspberry Pi OS Lite (64 Bit) auf Basis von Debian GNU/Linux Bullseye
- **Pi-hole**: Pi-hole v5.17.2 | FTL v5.23 | Web Interface v5.20.2
- **Fritz!OS**: 7.80
- **unbound**: 1.13.1 (Debian Bullseye Package)
- **Raspberry Pi**: Raspberry Pi 3 Mod. B

3. unbound

Wenn die im ersten Teil beschriebene Einrichtung abgeschlossen ist, kann nahtlos mit den folgenden Schritten fortgefahren werden. Die Inbetriebnahme von unbound ist schnell erledigt und besteht aus den folgenden Schritten: Installation, Erstellen einer Konfigurationsdatei, Anpassen der Pi-hole-Konfiguration und Testen des Setups. Die Installationsanleitung für unbound in der Pi-hole Dokumentation ist eine gute Vorlage. Ich habe mich im Folgenden daran orientiert und nur kleine Anpassungen vorgenommen.

3.1 Installation

Zunächst wird unbound mit einem Terminalbefehl installiert:

```
sudo apt install unbound
```

Da wir unbound über den Debian-Paketmanager installieren, wird die root.hints automatisch als Abhängigkeit des Pakets dns-root-data installiert. Die root.hints wird dann automatisch vom Paketmanager aktualisiert. Die root.hints wird benötigt, wenn eine Adresse/Domain nicht im Cache gespeichert ist. In diesem Fall muss unbound »ganz oben« am Ursprung beginnen und die Root-Nameserver befragen, um zu wissen, wo die Top-Level-Domain für die entsprechende Adresse/Domain zu finden ist. Wenn das Paket regelmäßig aktualisiert wird, ist kein manuelles Eingreifen erforderlich.

3.2 Konfiguration

Als Vorlage für die unbound-Konfiguration wird die Konfiguration des Pi-hole-Projekts verwendet. Diese zeichnet sich durch folgende Eigenschaften aus:

- Es werden nur DNS-Anfragen von der lokalen Pi-Hole-Installation beantwortet (auf Port 5335)
- Beantwortung von UDP- und TCP-DNS-Anfragen
- Überprüfung der DNSSEC-Signaturen und verwerfen von BOGUS-Domains
- Sicherheits- und Datenschutzoptimierungen
- Kompatibel mit IPv4- und IPv6-Netzwerk

Mit folgendem Befehl legen wir eine neue Konfigurationsdatei an:

```
sudo nano /etc/unbound/unbound.conf.d/pi-hole.conf
```

Kopiert den Inhalt und fügt ihn direkt in die eben angelegte Konfigurationsdatei ein:

```
server:
    # If no logfile is specified, syslog is used
    # logfile: "/var/log/unbound/unbound.log"
    verbosity: 0

    interface: 127.0.0.1
    port: 5335
    do-ip4: yes
    do-udp: yes
    do-tcp: yes

    # May be set to yes if you have IPv6 connectivity
    do-ip6: no

    # You want to leave this to no unless you have *native* IPv6. With 6to4
and
    # Terredo tunnels your web browser should favor IPv4 for the same
reasons
    prefer-ip6: no

    # Use this only when you downloaded the list of primary root servers!
    # If you use the default dns-root-data package, unbound will find it
```

```
automatically
    # root-hints: "/var/lib/unbound/root.hints"

    # Trust glue only if it is within the server's authority
    harden-glue: yes

    # Require DNSSEC data for trust-anchored zones, if such data is absent,
the zone becomes BOGUS
    harden-dnssec-stripped: yes

    # Don't use Capitalization randomization as it known to cause DNSSEC
issues sometimes
    # see
https://discourse.pi-hole.net/t/unbound-stubby-or-dnscrypt-proxy/9378 for
further details
    use-caps-for-id: no

    # Reduce EDNS reassembly buffer size.
    # IP fragmentation is unreliable on the Internet today, and can cause
    # transmission failures when large DNS messages are sent via UDP. Even
    # when fragmentation does work, it may not be secure; it is
theoretically
    # possible to spoof parts of a fragmented DNS message, without easy
    # detection at the receiving end. Recently, there was an excellent study
    #>>> Defragmenting DNS - Determining the optimal maximum UDP response
size for DNS <<<  # by Axel Koolhaas, and Tjeerd Slokker
(https://indico.dns-oarc.net/event/36/contributions/776/)
    # in collaboration with NLnet Labs explored DNS using real world data
from the
    # the RIPE Atlas probes and the researchers suggested different values
for
    # IPv4 and IPv6 and in different scenarios. They advise that servers
should
    # be configured to limit DNS messages sent over UDP to a size that will
not
    # trigger fragmentation on typical network links. DNS servers can switch
    # from UDP to TCP when a DNS response is too big to fit in this limited
    # buffer size. This value has also been suggested in DNS Flag Day 2020.
    edns-buffer-size: 1232

    # Perform prefetching of close to expired message cache entries
    # This only applies to domains that have been frequently queried
    prefetch: yes

    # One thread should be sufficient, can be increased on beefy machines.
In reality for most users running on small networks or on a single machine,
it should be unnecessary to seek performance enhancement by increasing num-
threads above 1.
    num-threads: 1

    # Ensure kernel buffer is large enough to not lose messages in traffic
```

```
spikes
    so-rcvbuf: 1m

    # Ensure privacy of local IP ranges
    private-address: 192.168.0.0/16
    private-address: 169.254.0.0/16
    private-address: 172.16.0.0/12
    private-address: 10.0.0.0/8
    private-address: fd00::/8
    private-address: fe80::/10


</file>
==== Hinweis ====

Alle unbound-Befehle (mit den Default-Werten) sind beim Debian-Projekt näher erläutert. Da auf dem Pi-hole (Version 5.17.1) in Kombination mit dem offiziellen Raspberry Pi OS Lite (64 Bit) immer noch ein Debian Bullseye läuft, müssen wir uns an der [[https://manpages.debian.org/bullseye/unbound/unbound.conf.5.en.html|dazugehörigen unbound.conf]] (unbound Version 1.13.1) für Bullseye orientieren.

Damit Pi-hole das Limit (1232) des EDNS Reassembly Buffers berücksichtigt, wird eine weitere Konfigurationsdatei erstellt:
<code>


sudo nano /etc/dnsmasq.d/99-edns.conf
```

Der Inhalt ist wie folgt:

```
edns-packet-max=1232
```

Danach ist die Konfiguration abgeschlossen und unbound kann gestartet werden. Wir werden dies jedoch auf einen späteren Zeitpunkt verschieben und noch weitere Konfigurationseinstellungen vornehmen. Ab hier weiche ich von der unbound Installationsanleitung des Pi-hole-Projekts ab.

3.3 IPv6: do-ip6

In der Konfiguration des Pi-hole-Projekts ist IPv6 standardmäßig deaktiviert. Das bedeutet aber nicht, dass eure IPv6 fähigen Clients keine IPv6 Domains mehr auflösen bzw. aufrufen können. Das funktioniert wunderbar. Der Parameter `do-ip6` steuert lediglich das Verhalten von unbound in Bezug auf IPv6:

> Enable or disable whether ip6 queries are answered or issued. Default is yes. If disabled, queries are not answered on IPv6, and queries are not sent on IPv6 to the internet nameservers. With this option you can disable the ipv6 transport for sending DNS traffic, it does not impact the contents of the DNS traffic, which may have ip4 and ip6 addresses in it.

Wenn man also möchte, dass unbound DNS-Anfragen über IPv6 an die Nameserver sendet, kann man diese Option aktivieren. Wenn zusätzlich die Option prefer-ip6 aktiviert ist, wird IPv6 gegenüber IPv4 beim Senden von DNS-Anfragen bevorzugt.

Ich persönlich habe beide Parameter der Standardkonfiguration unverändert gelassen, also IPv6 ausgeschaltet. Da diese beiden Parameter immer wieder für Verwirrung sorgen, wollte ich nur kurz darauf eingehen.

3.4 Erweiterung der Konfiguration

Die unbound-Konfiguration des Pi-hole-Projektes für den Pi-hole ist eher konservativ ausgelegt. Die Priorität liegt auf einer funktionierenden Konfiguration. Dies ist verständlich, da zusätzliche Befehle oder Änderungen das Verhalten von unbound negativ beeinflussen können. Die Folge: DNS-Auflösungen funktionieren womöglich nicht mehr sauber und Clients erhalten keine bzw. veraltete IP-Adressen.

Doch bevor ich meine Anpassungen vorstelle, möchte ich zunächst mit einem Irrglauben aufräumen: Viele glauben, dass die erste DNS-Anfrage an eine aufgerufene Domain etwas länger dauern kann, weil unbound die DNS-Anfrage über einen Root-Nameserver startet und sich bis zum zuständigen autoritativen Nameserver »durchfragen« muss, bevor der Domainname in die zugehörige IP-Adresse übersetzt wird. Dies ist zwar grundsätzlich richtig, aber die Annahme, dass eine Domain nach einmaliger Auflösung (Caching) zukünftig immer sofort verfügbar ist, ist falsch.

Jeder Resource Record (RR) einer Domain enthält eine Information, die als Time to live (TTL) bezeichnet wird. Für eine Domain gibt die TTL jedes Resource Records an, wie lange (in Sekunden) eine soeben erfolgte Namensauflösung voraussichtlich noch mindestens gültig bleibt. Während dieses Zeitraums kann das DNS-Caching verwendet werden. Nach Ablauf der TTL sollte der Client die entsprechende Namensauflösung verwerfen und bei Bedarf wiederholen. Das bedeutet: Sobald der TTL-Wert für eine Domain abgelaufen ist, wird unbound sich erneut bis zum autoritativen Nameserver »durchfragen«, um einen Domainnamen in die zugehörige IP-Adresse zu übersetzen. Die Annahme, dass also lediglich die »erste« DNS-Anfrage bei Aufruf einer Domain länger dauern kann, ist falsch. Nachfolgend ein Beispiel.

Mit dig können wir uns den TTL-Wert für eine Domain ausgeben lassen. Das funktioniert, wenn wir den autoritativen Nameserver einer Domain kennen:

```
dig +nocmd +noall +answer @root-dns.netcup.net www.kuketz-blog.de
```

Die Ausgabe ist wie folgt:

```
www.kuketz-blog.de.   86400   IN    CNAME    kuketz-blog.de.
kuketz-blog.de.   86400   IN    A        46.38.242.112
```

Der zweite Wert gibt den TTL-Wert der Domain zurück. Im Beispiel 86400. 86400 Sekunden sind 24 Stunden. Nach 24 Stunden wird unbound also den Cache verwerfen und die Domain erneut in die zugehörige IP-Adresse auflösen. Schauen wir uns eine weitere Domain an:

```
dig +nocmd +noall +answer @pns101.cloudns.net www.spiegel.de
```

Die Ausgabe ist wie folgt:

```
www.spiegel.de.   300     IN    CNAME    aacfb9d106f4.link11.de.
```

Der TTL-Wert der Spiegel-Domain beträgt 300 Sekunden, also 5 Minunten. Bereits nach 5 Minuten wird unbound erneut die IP-Adresse auflösen.

Glücklicherweise bietet unbound einige Parameter, um das Caching-Verhalten zu beeinflussen und den TTL-Wert für Domains zu überschreiben. Persönlich habe ich nachfolgende Befehle hinzugefügt, d.h. die Pi-hole-unbound-Konfiguration erweitert. Die Konfiguration läuft bei mir seit einigen Monaten reibungslos. Ich kann jedoch nicht vollständig ausschließen, dass die Ergänzungen möglicherweise Probleme verursachen werden. Daher möchte ich erneut darauf hinweisen, dass dieser Beitrag sich an fortgeschrittene Anwender richtet, die mit möglichen Problemen umgehen können.

Mit dem folgenden Befehl rufen wir die bereits erstellte Konfigurationsdatei auf:

```
sudo nano /etc/unbound/unbound.conf.d/pi-hole.conf
```

Anschließend werden die folgenden Parameter ergänzt:

```
## Performance
# More cache memory, rrset=msg*2 | Default: 4m, 4m
msg-cache-size: 32m
rrset-cache-size: 64m
# Time  to  live [minimum|maximum] for RRsets and messages in the cache |
Default: 0, 86400
cache-min-ttl: 3600
cache-max-ttl: 86400
# Serve old responses from cache with a TTL of 0 in the response without
waiting for the actual resolution to finish | Default: no, 0
serve-expired: yes
serve-expired-ttl: 86400
# Fetch DNSKEYs earlier (DNSSEC): More cpu usage, less latency | Default: no
prefetch-key: yes
# Helps to reduce the query rate towards targets that get a very high
nonexistent name lookup rate | Default: no
aggressive-nsec: yes

## Privacy | Default: no, no
hide-identity: yes
hide-version: yes
```

Auf einige Parameter möchte ich nachfolgend kurz eingehen, damit wir verstehen, was sie bewirken und wie dadurch die Domainauflösung für Clients beschleunigt werden kann:

- **prefetch [yes]**: Unbound versucht mit prefetch, einen Eintrag im Cache zu aktualisieren, nachdem die erste Antwort an den Client gesendet wurde. Wenn ein Client kurz vor dem Ende des TTL eine Anfrage für eine Domain stellt und ein zwischengespeicherter Eintrag vorhanden ist, wird dieser an den Client zurückgegeben. Gleichzeitig startet unbound die Auflösung der Domain und ermittelt die aktuelle IP-Adresse. Für oft besuchte Websites/Domains versucht

unbound sogar, den Cache auf dem neuesten Stand zu halten.
- **serve-expired [yes]**: Wenn diese Option aktiviert ist, versucht unbound, (alte) Informationen aus dem Cache auszuliefern, ohne auf den Abschluss der eigentlichen DNS-Auflösung zu warten. Die eigentliche Antwort landet später im Cache – unbound aktualisiert die Information also im Hintergrund. Wie lange unbound Informationen aus dem Cache ausliefert, wird mit dem Parameter `serve-expired-reply-ttl` beeinflusst.
- **serve-expired-ttl [86400]**: Begrenzt die Zustellung von abgelaufenen Antworten auf die konfigurierten Sekunden. Diese Option gilt nur, wenn `serve-expired` aktiviert ist. Ein empfohlener Wert gemäß RFC 8767 liegt zwischen 86400 (1 Tag) und 259200 (3 Tage).

Zusammengefasst kann man sich das wie folgt vorstellen: Unbound wird (abgelaufene) Antworten aus dem Cache priorisieren, den Cache für häufig besuchte/beliebte Domains auf dem neuesten Stand halten und für den Fall, dass ein abgelaufener Datensatz zugestellt werden muss (bspw. seltene Abfrage, Problem mit vorgelagerter Auflösung), sicherstellen, dass der Datensatz nicht älter als die angegebene Grenze (86400) ist.

Übrigens verwenden wir den Parameter cache-min-ttl (3600), um sicherzustellen, dass eine Domain einen TTL-Wert von mindestens 3600 Sekunden (60 Minuten) hat. Wenn das Minimum greift, werden die Daten länger als vom Domaininhaber beabsichtigt zwischengespeichert, sodass weniger Abfragen zum Nachschlagen der Daten erfolgen.

4. Anpassung Pi-hole

Nach der unbound-Konfiguration ist eine Anpassung über das Webinterface des Pi-hole erforderlich. Wir geben folgende Adresse in den Browser ein und melden uns an:

```
192.168.50.5/admin
```

Unter `Settings → DNS` nehmen wir folgende Einstellungen vor:

- Alle Häkchen (falls vorhanden) bei den DNS-Servern in der linken Ansicht entfernen
- Custom 1 (IPv4): **127.0.0.1#5335**
- Custom 3 (IPv6): **entfernen**

Upstream DNS Servers

IPv4		IPv6		Name
☐	☐	☐	☐	Google (ECS, DNSSEC)
☐	☐	☐	☐	OpenDNS (ECS, DNSSEC)
☐	☐			Level3
☐	☐			Comodo
☐	☐	☐	☐	DNS.WATCH (DNSSEC)
☐	☐	☐	☐	Quad9 (filtered, DNSSEC)
☐	☐	☐	☐	Quad9 (unfiltered, no DNSSEC)
☐	☐	☐	☐	Quad9 (filtered, ECS, DNSSEC)
☐	☐	☐	☐	Cloudflare (DNSSEC)

ECS (Extended Client Subnet) defines a mechanism for recursive resolvers to send partial client IP address information to authoritative DNS name servers. Content Delivery Networks (CDNs) and latency-sensitive services use this to give geo-located responses when responding to name lookups coming through public DNS resolvers. *Note that ECS may result in reduced privacy.*

Custom DNS servers (1 custom server enabled) −

The following list contains all DNS servers selected above. Furthermore, you can add your own custom DNS servers here. The expected format is one server per line in form of `IP#port`, where the `port` is optional. If given, it has to be separated by a hash `#` from the address (e.g. `127.0.0.1#5335` for a local `unbound` instance running on port `5335`). The port defaults to 53 if omitted.

```
127.0.0.1#5335
```

Anschließend wird unten auf der Seite `Use DNSSEC` angekreuzt.

Nach einem Klick auf `Save` ist euer Pi-hole in Kombination mit unbound fast einsatzbereit. Wir müssen nur noch eine Debian-Bullseye-Eigenheit anpassen.

5. Debian-Fix

Debian Bullseye+ Releases installieren automatisch ein Paket namens openresolv mit einer Konfiguration, die zu unerwartetem Verhalten in Kombination mit dem Pi-hole und unbound führt. Der unbound-resolvconf-Service weist resolvconf an, unbound als Nameserver in die Datei `/etc/resolv.conf` einzutragen – allerdings nur mit `127.0.0.1` ohne Angabe des benötigten Ports 5335. Lokale Dienste und Prozesse verwenden die Konfigurationsdatei, um den DNS-Server zu ermitteln. Der fehlende Port führt dazu, dass die DNS-Anfragen unbeantwortet bleiben. Dieser Konfigurationsfehler kann folgendermaßen behoben werden.

Der unbound-resolvconf-Service wird deaktiviert:

```
sudo systemctl disable --now unbound-resolvconf.service
```

Verhindert die Erstellung einer `resolvconf_resolvers.conf` für unbound, wenn resolvconf vom System/einem Prozess aufgerufen wird:

```
sudo sed -Ei 's/^unbound_conf=/#unbound_conf=/' /etc/resolvconf.conf
sudo rm /etc/unbound/unbound.conf.d/resolvconf_resolvers.conf
```

Danach haben wir es geschafft. Nach einem Neustart von unbound ist Pi-hole mit unbound einsatzbereit:

```
sudo service unbound restart
```

6. Funktionsprüfung

Wenn alle Arbeiten abgeschlossen sind, kann die Überprüfung erfolgen, ob die Kombination aus Pi-hole und unbound einwandfrei funktioniert. Zunächst wird geprüft, ob die DNSSEC-Valdierung funktioniert:

```
dig fail01.dnssec.works @127.0.0.1 -p 5335
```

Ausgabe:

```
; <<>> DiG 9.16.44-Debian <<>> fail01.dnssec.works @127.0.0.1 -p 5335
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 14820
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;fail01.dnssec.works.   IN    A

;; Query time: 0 msec
;; SERVER: 127.0.0.1#5335(127.0.0.1)
;; WHEN: Tue Oct 10 12:40:46 CEST 2023
;; MSG SIZE  rcvd: 48
```

Als Status wird ein SERVFAIL zurückgegeben und die IP-Adresse fehlt ebenfalls. Jetzt nochmal mit einer Domain, bei der DNSSEC korrekt validiert werden kann:

```
dig dnssec.works @127.0.0.1 -p 5335
```

Ausgabe:

```
; <<>> DiG 9.16.44-Debian <<>> dnssec.works @127.0.0.1 -p 5335
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 6033
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;dnssec.works.   IN    A

;; ANSWER SECTION:
dnssec.works.        21418    IN    A    5.45.107.88

;; Query time: 0 msec
;; SERVER: 127.0.0.1#5335(127.0.0.1)
;; WHEN: Tue Oct 10 12:43:44 CEST 2023
;; MSG SIZE  rcvd: 57
```

Als Status wird NOERROR und ebenso eine IP-Adresse (5.45.107.88) zurückgegeben. Das `ad-Flag` signalisiert, dass unbound die Antwort für authentisch hält bzw. die Validierung via DNSSEC funktioniert. Um zu verstehen, wie die DNSSEC-Validierung auf dem Pi-hole funktioniert und wie die Log-Einträge zustande kommen, empfiehlt sich ein Blick in den Artikel »Understanding DNSSEC validation using Pi-hole’s Query Log«.

Unbound beherrscht ebenfalls QNAME Minimisation (qname-minimisation [yes]) – das kann wie folgt geprüft werden:

```
dig txt qnamemintest.internet.nl +short @127.0.0.1 -p 5335
```

Ausgabe:

```
a.b.qnamemin-test.internet.nl.
```

```
"HOORAY - QNAME minimisation is enabled on your resolver :)!"
```

Von einem Linux-Client aus könnt ihr noch folgende Befehle für einen Test absetzen. Ihr müsst die IP-Adresse `192.168.50.5/fd00::a41a:eae7:9fd6:2f8c` mit der eures Pi-holes ersetzen:

- **IPv4 + UDP-Request**: dig @192.168.50.5 example.com +udp
- **IPv4 + TCP-Request**: dig @192.168.50.5 example.com +tcp
- **IPv4 + UDP-Request + IPv6-Domain**: dig @192.168.50.5 -t AAAA example.com
- **IPv6 + UDP-Request + IPv6-Domain**: dig -6 @fd00::a41a:eae7:9fd6:2f8c -t AAAA example.com

Ausgabe von `dig -6 @fd00::a41a:eae7:9fd6:2f8c -t AAAA example.com`:

```
; <<>> DiG 9.18.19-1~deb12u1-Debian <<>> -6 @fd00::a41a:eae7:9fd6:2f8c -t
AAAA example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35276
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;example.com.   IN    AAAA

;; ANSWER SECTION:
example.com.        86400    IN    AAAA
2606:2800:220:1:248:1893:25c8:1946

;; Query time: 232 msec
;; SERVER: fd00::a41a:eae7:9fd6:2f8c#53(fd00::a41a:eae7:9fd6:2f8c) (UDP)
;; WHEN: Tue Oct 10 13:09:44 CEST 2023
;; MSG SIZE  rcvd: 68
```

7. Logging

Standardmäßig protokolliert unbound Fehlermeldungen in der Protokolldatei `/var/log/syslog`. Insgesamt kennt unbound sechs verschiedene Logging-Stufen:

- Stufe 0 bedeutet keine Details, nur Fehler
- Stufe 1 liefert Betriebsinformationen
- Stufe 2 liefert detaillierte Informationen zum laufenden Betrieb
- Ebene 3 liefert Informationen auf Abfrage-Ebene
- Ebene 4 liefert Informationen auf Algorithmenebene
- Ebene 5 protokolliert Clients bei Cache-Fehlern

Sollte es aufgrund von Fehlern oder aus anderen Gründen notwendig sein, den Logging-Level zu erhöhen, so orientiert euch einfach an der Pi-hole-unbound-Konfiguration, das dort gut beschrieben ist.

8. Fazit

In diesem Artikel wurde gezeigt, wie man ein Pi-hole mit unbound konfiguriert. Damit wird einerseits die Abhängigkeit von öffentlichen DNS-Servern aufgehoben und andererseits werden DNS-Anfragen für Clients durch die Anpassung einiger unbound-Parameter (deutlich) schneller beantwortet. Diese Unabhängigkeit geht derzeit leider noch mit unverschlüsselten DNS-Anfragen (via UDP/TCP) einher, da die Root-Nameserver selbst Protokolle wie DNS over TLS (DoT) und auch DNS over HTTPS (noch) nicht unterstützen.

[Projekt, PiHole, Tipps & Tricks]