DNS | PiHole mit DoT+DNSSEC oder DoH

Was wollen wir?

Verschlüsselten DNS Verkehr mit DoT (DNS over TLS) oder DoH (DNS over HTTPS)

Warum wollen wir das?

Weil wir Wert auf Privatsphäre legen.

Und wie geht das genau?

Mit Stubby, Cloudflared & PiHole

Vorwort:

Viele hier haben ja eine PiHole Instanz am laufen, jedoch ist der DNS Verkehr in den meisten Fällen unverschlüsselt.

Das wollen wir nun ändern! Ich habe mich für DoT entschieden. Ich benutze Quad9, jedoch ist es natürlich möglich, sämtliche andere Resolver die DoT oder DoH unterstützen zu nehmen.

Ich habe bewusst auf die TLS Authentifizierung mit Schlüsseln verzichtet, da bei Servern mit Let's Encrypt Zertifikat diese regelmäßig aktualisiert werden müssen.

Die Wahrscheinlichkeit das hier Hijacking vorkommt ist schwindend gering, sollte jemand bedenken haben, kann er ja mit einem Auth-Key arbeiten.

Außerdem habe ich bewusst auf DoH verzichtet, weil DoH nicht im Trasport Layer sondern im Application Layer integriert ist - für mich ist DoT einfach richtiger "IMHO"

Trotzdem zeige ich in dem Tutorial auch den einfachen Weg mit DoH.

Das hier beschriebene Vorgehen bezieht sich auf die Installation auf einen RaspberryPi mit Raspbian Lite.

Bitte sucht euch die passenden Befehle für das Betriebssystem wo ihr PiHole am laufen habt raus, die Konfiguration ändert sich jedoch nicht!!!!

Bitte lest euch den ganzen Eintrag einmal durch, entscheidet dann welchen Weg ihr gehen wollt!

Tutorial | DoT+DNSSEC

Also los geht's - wir fangen mit DoT und DNSSEC an.

Erstmal das System auf den neusten Stand bringen:

```
sudo apt-get update && sudo apt-get dist-upgrade -y
```

Sollte PiHole noch nicht installiert sein:

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

Einfach dem Installationsassistenten folgen und einen Resolver deiner Wahl nehmen, wichtig dabei ist, dass der Pi oder Container hinterher immer die gleiche IP hat.

Nach der Installation passen wir dnsmasq an:

```
sudo nano /etc/dnsmasq.d/99-my-config.conf
```

Es muss folgende Zeile eingefügt werden:

```
listen-address=::1,127.0.0.1,XXX.XXX.XXX.XXX #(eure IP des PiHole)
```

Mit Strg+O, Enter, Strg+X Speichern

Jetzt installieren wir Stubby um die Voraussetzung für DoT+DNSSEC zu erlangen:

```
sudo apt install stubby
```

Jetzt stoppen wir Stubby und passen die Konfiguration an:

```
sudo systemctl stop stubby
sudo nano /etc/stubby/stubby.yml
```

❗ ACHTUNG: YML-Dateien reagieren extrem sensitiv auf Leertasten u.ä. also kontrolliert eure Config! Außerdem sind das hier nur Auszüge aus der config. Ihr könnt nicht einfach den Code mit Copy&Paste übernehmen. Ihr müsst euch hier step by step durcharbeiten. ❗

Nun passen wir ein paar Einstellungen an:

Code

```
resolution_type:                                GETDNS_RESOLUTION_STUB
dns_transport_list:
-                                                 GETDNS_TRANSPORT_TLS
tls_authentication:                          GETDNS_AUTHENTICATION_REQUIRED
tls_query_padding_blocksize:                                         128
round_robin_upstreams: 1
```

Bei `round_robin_upstreams` könnt ihr auch die 0 setzen, solltet Ihr nur einen Resolver benutzen. Wenn Ihr mehrere Resolver nutzt dann setzt die 1 und Stubby fragt immer zufällig an.

Jetzt können wir uns entscheiden, ob wir ECS aktivieren oder nicht. Bei mir ist es deaktiviert und sämtliches Streaming funktioniert einwandfrei.

Sollte es bei euch also im Nachgang Probleme mit dem Streaming geben, dann solltet ihr ECS aktivieren.

Wir deaktivieren es nun erstmal und passen weiter die config an:

Code

```
edns_client_subnet_private                    :                      1
idle_timeout:                                                    10000
listen_addresses:
-                                                        127.0.2.2@10053
dnssec: GETDNS_EXTENSION_TRUE #hiermit aktivieren wir DNSSEC
```

Ziemlich weit unten in der config, findet ihr Optional Upstream - ich benutze Quad9.

Code

```
##  Quad  9  'secure'  service  -  Filters,  does  DNSSEC,  doesn't  send  ECS
-                              address_data:                                9.9.9.9
tls_auth_name:                                                        "dns.quad9.net"
##  Quad  9  'secure'  service  -  Filters,  does  DNSSEC,  doesn't  send  ECS
-                           address_data:                                2620:fe::fe
tls_auth_name:                                                        "dns.quad9.net"
##             Cloudflare             1.1.1.1             and             1.0.0.1
#                                       -                   address_data:
#                                                                                  tls_a
#                                       -                   address_data:
#                                                                                  tls_a
##                                Cloudflare                                servers
#                               -               address_data:                2606:4700:4700::1
#                                                                                  tls_a
#                               -               address_data:                2606:4700:4700::1
#    tls_auth_name: "cloudflare-dns.com"
```

Alles anzeigen

Wir wählen unseren Wunsch-Resolver und lassen die anderen aus kommentiert.

Ihr solltet also die # vor eurem favorisierten Resolver entfernen.

Jetzt speichern wir die config ab mit Strg+O, Enter, Strg+X

Als nächstes sorgen wir dafür, dass Stubby automatisch nach einem Absturz neu gestartet wird:

```
sudo nano /lib/systemd/system/stubby.service
```

Hier passen wir folgendes an:

Code

```
Restart=on-failure
RestartSec=1
```

Jetzt speichern wir die config ab mit Strg+O, Enter, Strg+X

Jetzt führen wir folgende Befehle aus:

```
sudo systemctl daemon-reload

sudo systemctl restart stubby
```

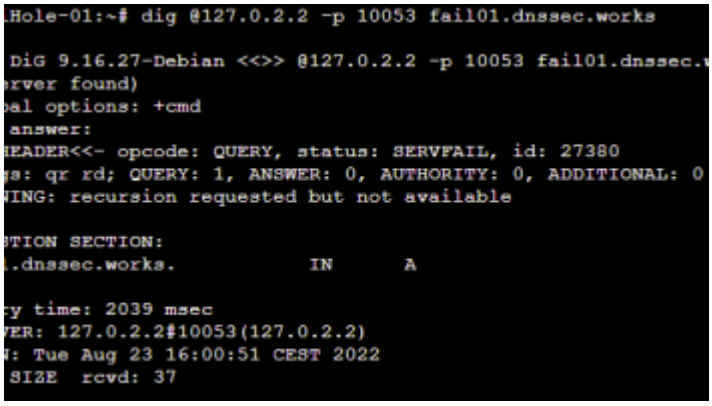
Jetzt installieren wir dnsutils:

```
sudo apt install dnsutils
```

Jetzt testen wir ob DNSSEC funktioniert.

```
dig @127.0.2.2 -p 10053 fail01.dnssec.works
```

Die Antwort sollte SERVFAIL lauten. Hiermit haben wir validiert, dass DNSSEC funktioniert.



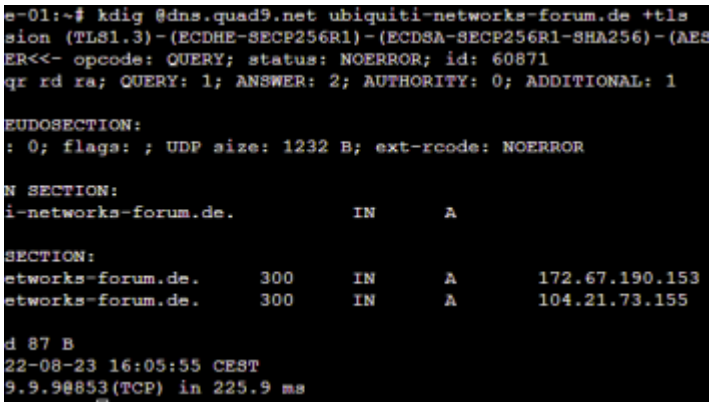
Jetzt installieren wir knotdnsutils:

```
sudo apt install knot-dnsutils
```

Nun testen wir ob TLS funktioniert (solltet ihr einen anderen Resolver benutzen, bitte "dns.quad9.net" durch den Resolver eurer Wahl ersetzen):

```
kdig @dns.quad9.net ubiquiti-networks-forum.de +tls
```

Nun sollte in der obersten Zeile der Antwort etwas von TLS stehen. Hiermit haben wir validiert, dass TLS funktioniert.



Da wir in der Stubby Config festgelegt haben, dass Anfragen ausschließlich über TLS ohne Fallback gesendet werden sollen, haben wir nun alle Voraussetzungen für DoT erfüllt.

Nun gehen wir auf die PiHole Weboberfläche

Unter Settings -> DNS, deaktivieren wir nun alle vorgegebenen Upstream DNS Server

Jetzt tragen wir bei Custom Upstream folgendes ein:

`127.0.2.2#10053`

Speichern und den Resolver neu starten.

Wichtig ist, dass der Haken bei DNSSEC nicht gesetzt wird, sonst kommt in den Logs die Meldung Insecure.

Das hat den Hintergrund, dass PiHole bei Stubby anfragt, und diese Anfrage kein DNSSEC validiert. Erst die Anfrage durch Stubby an den Resolver wird validiert.

Nun tragen wir bei all unseren LANs und WANs als DNS Adresse den PiHole ein (zuvor konfiguriert)

Geschafft, PiHole macht jetzt DNS mit DoT und DNSSEC

Ich für meinen Teil gebe mich mit dem einen Resolver und DoT+DNSSEC zufireden. ich Brauche keinen Mix aus DoH und DoT.

Tutorial | DoH

Erstmal das System auf den neusten Stand bringen:

```
sudo apt-get update && sudo apt-get dist-upgrade -y
```

Sollte PiHole noch nicht installiert sein:

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

Einfach dem Installationsassistenten folgen und einen Resolver deiner Wahl nehmen, wichtig dabei ist, dass der Pi oder Container hinterher immer die gleiche IP hat.

Jetzt installieren wir Cloudflared:

For Debian/Ubuntu

Code

```
wget      https://bin.equinox.io/c/VdrWdbjqyF/cloudflared-stable-linux-amd64.deb
sudo        apt-get         install         ./cloudflared-stable-linux-amd64.deb
cloudflared -v
```

For CentOS/RHEL/Fedora

Code

```
wget      https://bin.equinox.io/c/VdrWdbjqyF/cloudflared-stable-linux-amd64.rpm
sudo         yum          install          ./cloudflared-stable-linux-amd64.rpm
cloudflared -v
```

#arm64 architecture (64-bit Raspberry Pi)

Code

```
wget https://github.com/cloudflare/cloudflared/releases/latest/download/cloudflared-linux-arm64
sudo                mv                cloudflared                /usr/local/bin
sudo              chmod              +x              /usr/local/bin/cloudflared
cloudflared -v
```

#armhf architecture (32-bit Raspberry Pi)

Code

```
wget         https://bin.equinox.io/c/VdrWdbjqyF/cloudflared-stable-linux-arm.tgz
tar                    -xvzf                     cloudflared-stable-linux-arm.tgz
sudo                cp                ./cloudflared                /usr/local/bin
sudo               chmod              +x              /usr/local/bin/cloudflared
cloudflared -v
```

Wir legen einen User für Cloudflared an:

```
sudo useradd -s /usr/sbin/nologin -r -M cloudflared
```

Nun passen wir die config an:

```
sudo nano /etc/default/cloudflared
```

Tragt hier nun euren gewünschten Resolver ein.

```
# Commandline args for cloudflared, using Cloudflare DNS
CLOUDFLARED_OPTS=--port 11053 --upstream https://9.9.9.9/dns-query
```

Mit Strg+O, Enter, Strg+X Speichern

Jetzt aktualisieren wir die Berechtigung:

```
sudo chown cloudflared:cloudflared /etc/default/cloudflared
sudo chown cloudflared:cloudflared /usr/local/bin/cloudflared
```

Da Cloudflared automatisch starten soll passen wir nun den Service an:

```
sudo nano /etc/systemd/system/cloudflared.service
```

Mit Strg+O, Enter, Strg+X Speichern

Eure Config sollte so aussehen

Code

```
[Unit]
Description=cloudflared DNS over HTTPS proxy
After=syslog.target network-online.target

[Service]
Type=simple
User=cloudflared
EnvironmentFile=/etc/default/cloudflared
ExecStart=/usr/local/bin/cloudflared proxy-dns $CLOUDFLARED_OPTS
Restart=on-failure
RestartSec=10
KillMode=process

[Install]
WantedBy=multi-user.target
```

Alles anzeigen

Mit Strg+O, Enter, Strg+X Speichern

Jetzt starten wir den Service:

Code

```
sudo systemctl enable cloudflared
sudo systemctl start cloudflared
sudo systemctl status cloudflared
```

Nun gehen wir auf die PiHole Weboberfläche

Unter Settings -> DNS, deaktivieren wir nun alle vorgegebenen Upstream DNS Server

Jetzt tragen wir bei Custom Upstream folgendes ein:

```
127.0.0.1#11053
```

Speichern und den Resolver neustarten.

Nun tragen wir bei all unseren LANs und WANs als DNS Adresse den PiHole ein (zuvor konfiguriert)

Geschafft, PiHole macht jetzt DNS mit DoH

Schlusswort:

Wie bereits oben beschrieben ist es möglich, einen Mix aus DoH und DoT zu verwenden.

Somit könnt Ihr auch beide Resolver in den Upstreams eintragen. Aber wie oben schon gesagt, liegt DoT auf dem Transportlayer. DoH ist Applikationsseitig. Deswegen ist meine Entscheidung: **DoT "IMHO"**

Trotz meiner Entscheidung und Meinung ist DoH immer noch besser als kein verschlüsseltes DNS!

Entscheidet euch für euren Favorit! Informiert euch über das Thema!

Mir ist außerdem aufgefallen, dass PiHol eigentlich seltensten Falles auf den zweiten Resolver ausweicht, somit macht im PiHole eher nur ein Upstream Sinn.

Sollte hier etwas missverständlich oder unverständlich beschrieben sein, bitte meldet euch bei mir, damit ich dies direkt richtig stellen bzw. anpassen kann.

#PiHole #DoT #DoH #Security #Stubby #Cloudflared

Auswählen:

Gültige Software-Version Keine Firmware-Relevanz!