

# lokale Zertifikate

in der heutigen Zeit kommt man sehr bequem und schnell zu einem kostenlosen [SSL](#) Zertifikat mit [Let's Encrypt](#). Leider gilt das aber nur für öffentliche, registrierte Domänen. Für das Heimnetzwerk ist das leider keine wirkliche Lösung. Deshalb möchte ich hier beschreiben / aufzeichnen, wie man SSL für sein Heimnetzwerk implementieren kann.

- [Was passiert bei Browser Anfragen?](#)
- [Voraussetzungen / Empfehlungen](#)
- [Zertifikate erstellen](#)
- [Zertifikate verteilen](#)
- [DNS Resolver](#)
- [Gutzi](#)

# Was passiert bei Browser Anfragen?

DNS Namen werden von Browsern von rechts nach links gelesen.  
z.B.

- `http://server.example.com` -> `com` - `example` - `server` - Protokoll

“ [Definition](#) einer Domäne: **Top Level Domain** (TLD) - **Second Level Domain** (SLD)  
- Subdomain (optional) - Protokoll

Damit wird sichergestellt, dass zuerst lokal gesucht wird und erst im Fall eine Information ist nicht lokal vorhanden, eine Auflösungsanfrage zu einem "Resolver" weitergegeben wird.

Das wird in lokalen Heimnetzen grundsätzlich auf jedem Geräte mit der Datei `"/etc/hosts"` gelöst ( Microsoft: `C:\Windows\System32\drivers\etc\hosts` ).

***Bei erhöhter Anzahl von Geräten im Heimnetzwerk wird dieses Verwalten mit der Zeit ineffizient und fehleranfällig!***

Deshalb werden bei Bedarf in Heimnetzwerken auch lokale DNS Resolver (ein Gerät, das über DHCP oder durch manuellem Eintrag auf einem Geräte bekannt gegeben wird eingesetzt (typische Vertreter: `bind9`, `dnsmasq`, `PI-Hole`, `AdGuard`, `Technitium` und ganz sicher noch jede Menge andere). Hier trennt sich jetzt die Spreu vom Weizen... Man muss den DNS Teil der Tools beherrschen und wissen was man tut! Bis jetzt hat das alles noch nichts mit `http://` oder `https://` zu tun!

## `http://` - `https://` Auflösung

Wir wollen uns ja auf SSL = `https://` konzentrieren! Wenn der Browser o.a. Thematik abgearbeitet hat, schaut er ob ein entsprechender Web Server (z.B. `apache2`, `Nginx`, andere...) eine aufrufbare Seite zur Verfügung stellt. **UND** jetzt wird es interessant!!!! Der Browser geht nämlich aufgrund des vom Web Server hinterlegtem Zertifikat lokal schauen / abfragen ob eine vertrauenswürdige Stammzertifizierungsstelle hinterlegt ist! Und das lässt das Heimnetzwerk mit `https` & Domänenname scheitern! Ihr findet die Standard Stammzertifizierungsstellen für:

- MAC : `/Library/Keychains/System.keychain`
- Linux : `/usr/local/share/ca-certificates`

- Windows : certmgr.msc

Aber natürlich nicht für den geneigten Leser! Wir nehmen das an die Hand und realisieren das Konzept für ein Heimnetzwerk!

# Voraussetzungen / Empfehlungen

## 1. Hardware / Betriebssystem

Ich würde euch empfehlen einen Raspberry PI 4 oder 5 anzuschaffen und den PI als 24h / 7T Server im Heimnetzwerk laufen zu lassen. Damit habt ihr eine weitgehendst kostenminimale, verlässliche Hardware am Start. Ein Betrieb mit LAN (WLAN würde ich nicht empfehlen!) und "ssh" ist ausreichend. ***Ein Windows basierendes Gerät würde ich persönlich nicht für diesen Zweck einrichten (beschreibe ich auch nicht)!***

## 2. Software

- zur Zeit ist das Raspberry OS basierend auf Debian 12 (Bookworm)
- ich würde euch das OS Lite 64-bit empfehlen
- Ihr braucht "[openssl](#)", welches im Standard PI OS verfügbar ist

## 3. zusätzliche Software

Da die SSL Implementierung so eng mit DNS Mechanismen verknüpft ist würde ich euch auch einen DNS Resolver ans Herz legen. Empfehlung: [Technitium](#)! Sollten jetzt die PI-Hole & AdGuard Fans den Kopf schütteln, schaut bitte mal dieses Video:

<https://www.youtube.com/embed/O1hBQSIV3ts>

**Obwohl das Terminal strapaziert wird, werde ich alles in deutscher Sprache beschreiben. Die Kommandos sind nun mal in "English"!**

## 4. Entscheidet euch für einen internen Domänenname

Typischerweise könnt ihr, falls ihr eine AVM Fritz Box betreibt eine Domäne "fritz.box" realisieren. Dann wären die Namen. z.B. nas.fritz.box, router.fritz.box, repeater.fritz.box, etc. Allerdings spiegelt das nicht unbedingt euer Heimnetz wieder. Es wäre also eine Namensgebung, wie ein eindeutiger, selbst sprechender Name, z.B. zuhause,lan oder thomas.home, besser geeignet.

***Bitte verwende keine .local Domäne, da .local von lokalen mDNS-Diensten verwendet wird und bei einigen Geräten zur "Verwirrung", sprich nicht Erreichbarkeit oder eigenartiges Verhalten im Netzwerk führen kann!***

# Zertifikate erstellen

“ Alle nachfolgenden, erforderlichen Eingaben, Kommandos, etc. sind im Terminal des PI OS erforderlich!

Je nach dem welche Anleitung ihr im Internet schon gelesen habt, werden immer verschiedene Verzeichnisse zur Ablage der Zertifikate vorgeschlagen oder verwendet. Ich würde euch vorschlagen ein Verzeichnis zu erstellen, das ihr immer verwendet um eure eigenen Zertifikate zu erstellen, zu bearbeiten und als Speicherort in den Browser 'VirtualHost' Blöcken anzugeben.

Für unser Beispiel verwenden wir:

- TLD = lan
- SLD = zuhause
- subdomain = nas

## 1. erstellen einer vertrauenswürdigen Stammzertifizierungsstelle

### 1.1 erstellen eines Privaten Schlüssels für die Stammzertifizierungsstelle

Kommando: "openssl genrsa -des3 -out <SLD, ohne TLD>.key 2048"

- z.B. euer Domänenname = zuhause.lan, dann nur "zuhause"!
- ihr werdet dann nach einer "Pass Phrase" gefragt **8-tung!** vergebt bitte eine Passwort, dass ihr **nicht vergesst!**, da ansonsten später massive Probleme auftauchen können, da ihr dieses Passwort immer mal wieder braucht, je nachdem wo und wie ihr euer Zertifikat gebrauchen wollt!

- “ *noch ein Tipp:*

  - je länger das Passwort, desto besser die Verschlüsselung. Obwohl ich auch nicht übertreiben würde, da ja sich alles nur in eurem Heimnetzwerk abspielt.
  - übrigens: falls ihr im Laufe der Arbeiten feststellt, das ihr einen anderen TLD oder SLD benutzen wollt, müsst ihr immer wieder von vorne anfangen. Zertifikate kann man nicht ändern!

### 1.2 erstellen des Stammzertifikats

- Kommando: "openssl req -x509 -new -nodes -key <SLD>.key -sha256 -days 3650 -out >SLD>.pem"
- es werden jetzt noch folgenden zusätzliche Informationen abgefragt:
- 

```

“
Country Name (2 letter code): <>
State or Province Name (full name): <>
Locality Name (eg, city): <>
Organization Name (eg, company): <>
Organizational Unit Name (eg, section): <>
Common Name (e.g. server FQDN or YOUR name): <SLD.TLD>
Email Address: <>

```

<> = könnt ihr ausfüllen, muss man aber nicht!

## 2. erstellen von Zertifikaten basierend auf dem Stammzertifikat

“ Diesen Ablauf müsst ihr für jede subdomain (meistens Geräte, z.b. NAS) wiederholen!

### 2.1 erstellen eines Privaten Schlüssels für das Zertifikat

Kommando: "openssl genrsa -out <subdomain.SLD.TLD>.key 2048"

- unser Beispiel = nas.zuhause.lan

### 2.2 erstellen eines CSR (Code Signing Request)

Kommando: "openssl req -new -key <subdomain.SLD.TLD>.key -out <subdomain.SLD.TLD>.csr"

- unser Beispiel = nas.zuhause.lan

### 2.3 erstellen einer X509 V3 Zertifikatserweiterungs-Konfigurationsdatei

Kommando: "nano <subdomain.SLD.TLD>.ext"

```

authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]

```

- unser Beispiel = nas.zuhause.lan

## 2.4 das Zertifikat erstellen

(mit unserer CSR, dem privaten Schlüssel der CA, dem CA-Zertifikat und der Konfigurationsdatei)

Kommando: "openssl x509 -req -in <subdomain.SLD.TLD>.csr -CA <SLD>.pem -CAkey <SLD>.key -CAcreateserial -out <subdomain.SLD.TLD>.crt -days 3650 -sha256 -extfile <subdomain.SLD.TLD>.ext"

- unser Beispiel = SLD: zuhause - subdomain.sld.TLD: nas.zuhause.lan

“ hey, ihr "Durchhalter...", ihr habt es tatsächlich geschafft... na, ja - fast! ☐☐ Das Stammzertifikat muss ja noch verteilt werden, damit die Browser euer Zertifikat erfolgreich validieren können und der Web Server braucht ja noch die Information welches Zertifikat er verwenden soll, aber dann, versprochen, "klappt's auch mit dem Nachbarn" ☐☐



# Zertifikate verteilen

## 1. Stammzertifikat

“ Das Stammzertifikat muss auf alle Geräte installiert werden, von denen ihr auf euren Web Server zugreifen / browsen wollt! Es wäre deshalb ratsam euer "Zertifikatsverzeichnis" auf den jeweiligen Geräten als Laufwerk verbunden zu haben!

### 1.1 MAC

Kommando: "sudo security add-trusted-cert -d -r trustRoot -k  
"/Library/Keychains/System.keychain" <Verzeichnis><SLD>.pem

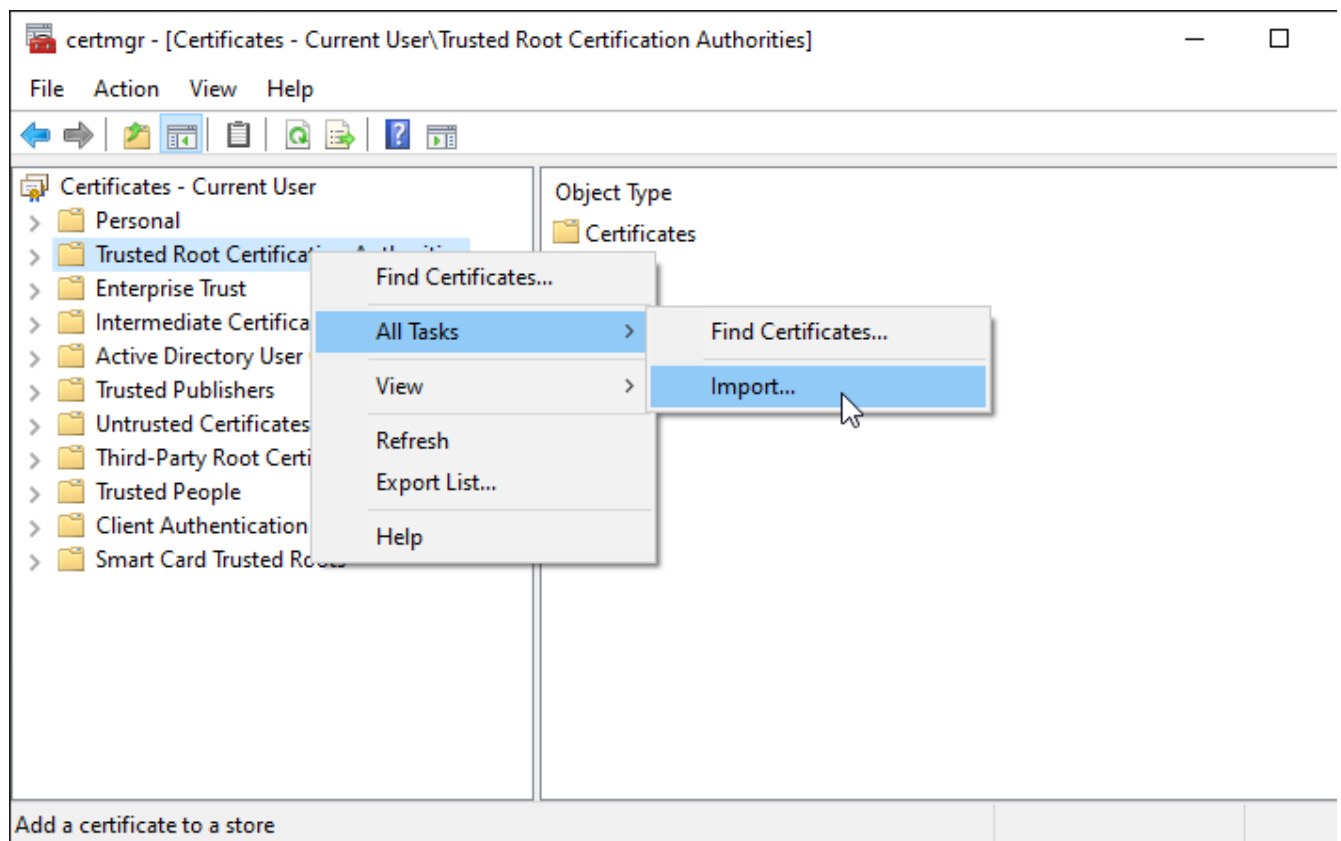
### 1.2 Linux

Kommando: "sudo cp <Verzeichnis><SLD>.pem /usr/local/share/ca-certificates/<SLD>.crt"

Kommando: "sudo update-ca-certificates"


### 1.3 Windows

Kommando: "WIN + R - Eingabe: certmgr.msc" - OK



NEXT

×

←  Certificate Import Wizard

**File to Import**  
Specify the file you want to import.

---

File name:

Note: More than one certificate can be stored in a single file in the following formats:  
Personal Information Exchange - PKCS #12 (.PFX, .P12)  
Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)  
Microsoft Serialized Certificate Store (.SST)

↗

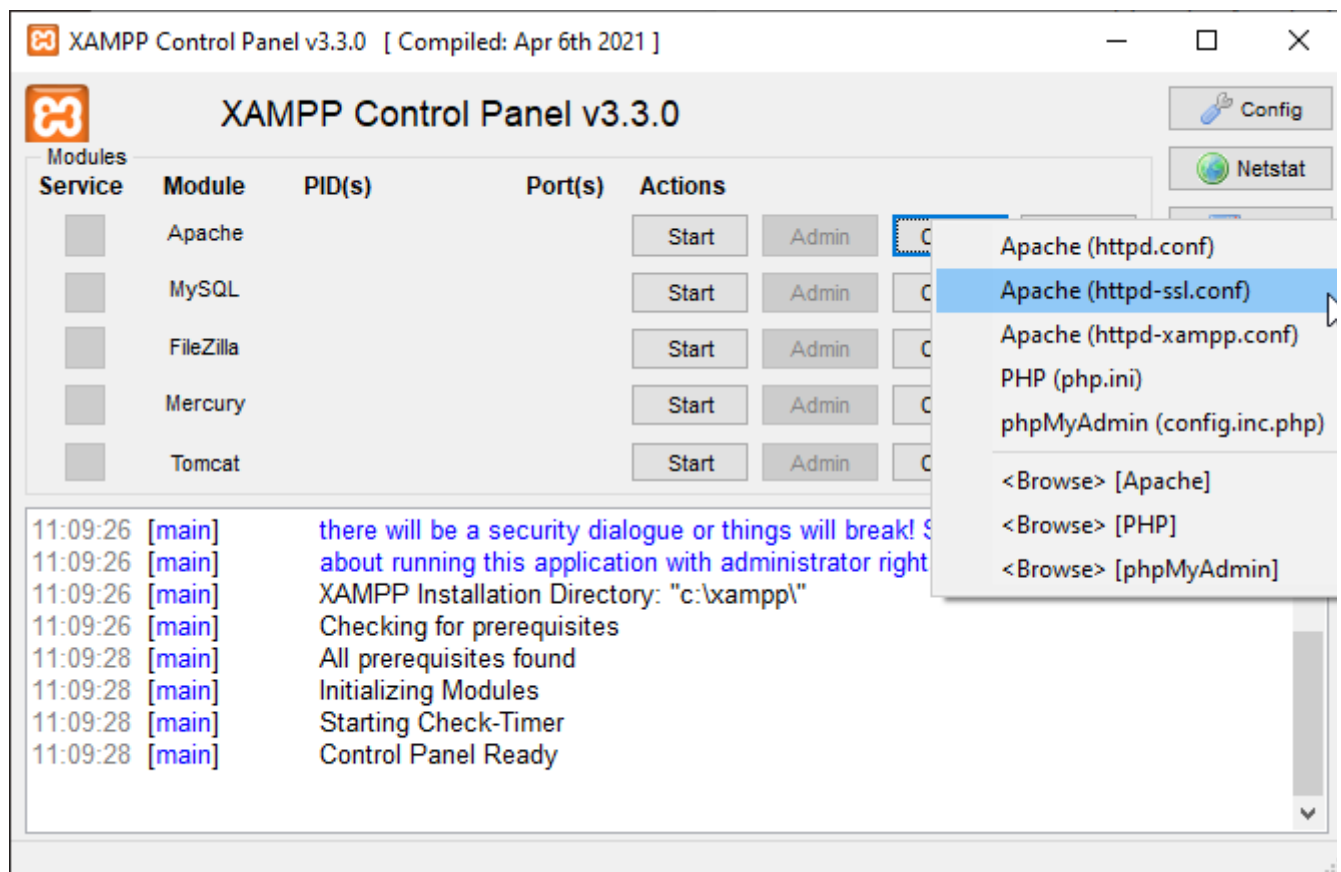
hoffe, den Rest bekommt ihr selber "zusammen geklickt" ! ☐☐

## 2. Web Server Zertifikat

### 2.1 Apache

“ Stellt bitte sicher, das ihr "a2enmod ssl" angeschaltet habt!

- geht in das Apache Verzeichnis "sites-available" und erstellt eine .conf Datei (geht natürlich auch mit der default-ssl.conf). Solltet ihr mit XAMPP den Web Server betreiben, dann:



- ändert die VirtualHost Direktive

```
<VirtualHost *:443>
    ServerName <>
    DocumentRoot <>

    SSLEngine on
    SSLCertificateFile <Verzeichnis>/<subdomäne.SLD.TLD>.crt
    SSLCertificateKeyFile <Verzeichnis>/<subdomäne.SLD.TLD>.key
</VirtualHost>
```

- Kommando: "a2ensite <der Dateiname>.conf"
- Kommando: "service apache2 restart"

“ im Fall von XAMPP Apache stoppen und wieder starten!

## 2.1 NGINX

```
server {
    listen          443 ssl;
    server_name     <subdomäne.SLD.TLS>;
```

```
ssl_certificate      <Verzeichnis><subdomäne.SLD.TLS>.crt;  
ssl_certificate_key  <Verzeichnis><subdomäne.SLD.TLS>.key;  
ssl_protocols        TLSv1.2 TLSv1.3;  
ssl_ciphers          HIGH:!aNULL:!MD5;  
...  
}
```

“ Jetzt habt es ihr aber geschafft!!!! ☐ Versteht bitte, das es hier um lokale Zertifikate gegangen ist und nicht wie man eine PI generiert oder wie man einen Webserver aufsetzt oder, oder, oder... Es bliebe nur noch die eigentliche Randthematik DNS Resolver übrig um das Bild abzurunden. Selbst wenn alles richtig gemacht wurde, kann es euch passieren, das eure Domäne nicht ansprechbar in euren Browsern ist.... ☐ Deshalb noch ein Abschlusskapitel DNS Resolver! Hey, und natürlich auch ein "Gutzi" fürs durchhalten. Schaut doch mal unter "Gutzi"!!!! ☐

# DNS Resolver

“ Um eure SSL Umgebung wirklich und vollumfänglich nutzen zu können braucht ihr einen lokale DNS Resolver der euer Heimnetzwerk kennt...

Es gibt hier verschiedene Möglichkeiten diese Technologie (DNS) auch mit "Hauskenntnissen" erfolgreich zu realisieren.

## 1. hosts Datei

Ihr könnt euere Geräte in einer hosts Datei definieren und z.B. das in alle hosts Dateien der betroffenen Geräte manuell verteilen. Das kann aber schnell in zeitintensive Arbeit ausarten und ist fehleranfällig! Des weiteren gibt es Geräte (z.B. IOT, Netzwerkgeräte, TVs) die gar keine manuelle hosts Verwaltung anbieten.

## 2. Resolver

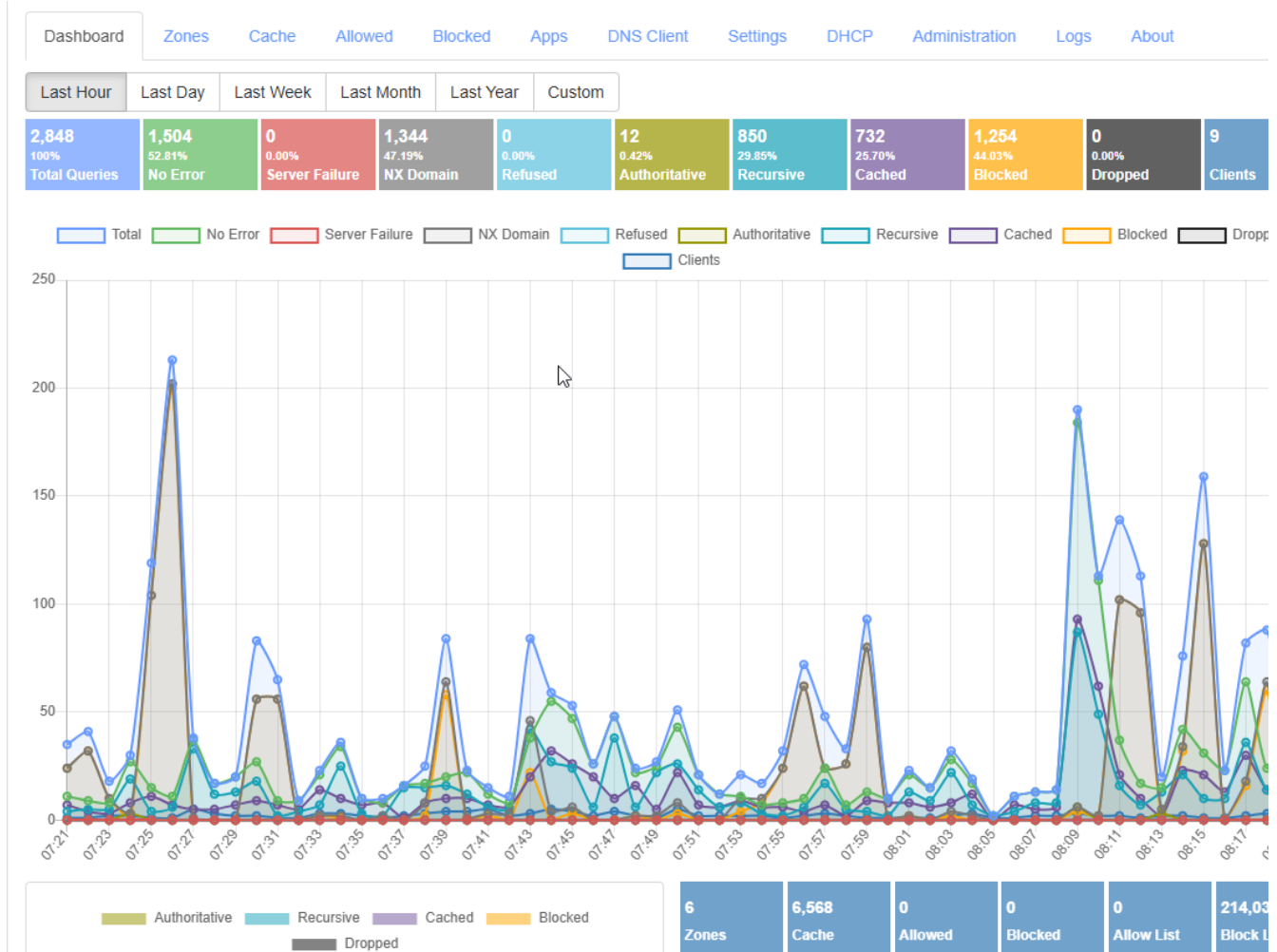
Hier kämen die Standard Tools wie z.B. bind9, dnsmasq in Frage. Die Verfügbarkeit ist allerdings je nach Betriebssystem fraglich. Außerdem ist meistens ein "Scheitern" vor programmiert, da man die Technologie und die Tools , na sagen wir mal, semi-professionell, beherrschen sollte.

## 3. Hybrid Applikationen

Hier wären die bekanntesten Tools wohl PI-Hole und AdGuard. Bedenkt bitte, das diese Tools aus einem anderen Grund endstanden sind: Ad Blocker!!!!!! Bedingt durch das Konzept (Fundament, Aufbau) wird es für die Tools immer schwieriger den modernen DNS Konzepten zu folgen, soll heißen die Technologien (z.B. DNSSEC) mit allen, möglichen Methoden (z.B. DNS-over-HTTPS oder DNS-over-QUIC) zu implementieren.

Eine etwas momentan noch weniger bekannte aber umso mehr bemerkenswerte Applikation ist [Technitium](#). Es vereint alle "Heim Informatiker" Ansprüche an eine moderne, anpassbare, nachhaltige DNS Resolver Lösung. Es kann Ad Blocking mit Anspruch auf Flexibilität und individuelle Anpassbarkeit, eine wirklich einfach einzurichtende und wartbare Resolver Implementierung wie eine, wenn gewünscht, extensive Analyse und Berichtsumgebung. Hm, einziger Nachteil könnte die Applikationssprache sein: English. Allerdings ist das sehr technisch und da ist English alle mal besser als eventuelle verunglückte deutsch Übersetzungen. Auch hilf das verlinkte Video sehr gut!!!!

### **Dashboard**



## DNS Resolver

DNS Server - netfactory

Dashboard

Zones

Cache

Allowed

Blocked

Apps

DNS Client

Settings

DHCP

Administration

Logs

About

← Back

hopto.home

Primary

Enabled

Add Record

Disable Zone

Delete Zone

Options

Permissions

DNSSEC

Name

abc or a\* or \*b\* or a?c

Type

Page Number

1

Records Per Page

10

1-6 (6) of 6 records (page 1 of 1)

| # | Name       | Type | TTL       | Data   |
|---|------------|------|-----------|--|
| 1 | @          | NS   | 3600 (1h) | <div><div>Name Server: netfactory</div><div>Last Used: 0001-01-01 00:00:00 (never)</div><div>Last Modified: 2025-06-15 08:55:21 (3 days ago)</div></div> <div><div>Edit</div><div>Disable</div><div>Delete</div></div>   |
| 2 | @          | SOA  | 900 (15m) | <div><div>Primary Name Server: netfactory</div><div>Responsible Person: hostadmin@hopto.home</div><div>Serial: 5</div><div>Refresh: 900 (15m)</div><div>Retry: 300 (5m)</div><div>Expire: 604800 (1w)</div><div>Minimum: 900 (15m)</div><div>Use Serial Date Scheme: false</div><div>Last Used: 2025-06-18 08:12:44 (10 minutes ago)</div><div>Last Modified: 2025-06-15 08:55:21 (3 days ago)</div></div> <div><div>Edit</div><div>Disable</div><div>Delete</div></div> |
| 3 | netfactory | A    | 3600 (1h) | <div><div>192.168.158.10</div><div>Last Used: 2025-06-18 08:12:44 (10 minutes ago)</div></div> <div><div>Edit</div><div>Disable</div><div>Delete</div></div>   |

## Ad Blocking



Blocking Answer TTL

30

seconds (default 30)

The TTL value in seconds that must be used for the records in a blocking response. This is the TTL value that the client will use to cache the blocking response.

Allow / Block List URLs

https://raw.githubusercontent.com/StevenBlack/hosts/master/alternates/fakenews-s-gambling-social/hosts

Quick Add

None

Default

Steven Black [adware + malware]

Steven Black [adware + malware + fakenews]

Steven Black [adware + malware + gambling]

Steven Black [adware + malware + porn]

Steven Black [adware + malware + social]

Steven Black [adware + malware + fakenews + gambling]

Steven Black [adware + malware + fakenews + porn]

Steven Black [adware + malware + fakenews + social]

Steven Black [adware + malware + gambling + porn]

Steven Black [adware + malware + gambling + social]

Steven Black [adware + malware + porn + social]

Steven Black [adware + malware + fakenews + gambling + porn]

Steven Black [adware + malware + fakenews + gambling + social]

Steven Black [adware + malware + fakenews + porn + social]

Steven Black [adware + malware + gambling + porn + social]

Steven Black [adware + malware + fakenews + gambling + porn + social]

OIDS Big [Adblock Plus]

Block List Update Interval

Block List Next Update On

Notel DNS Server will use the data returned to it from the file containing list of domains to block, wildcard characters are not allowed.  
[Help: Blocking Internet Ads Using DNS Sinkhole](#)

Save Settings

Flush Cache

Backup Settings

Restore Settings

“meine Empfehlung: Technitium

# Gutzi

“Je nach Größe eures Heimnetzwerkes müssen die Schritte "[Zertifikate erstellen](#)" dauernd abgearbeitet werden, was nervig werden kann. Vor allen Dingen wenn man sich mal nach einiger Zeit für einen neuen Domänen Namen entscheiden will... Es gibt auch Geräte, z.B. AVM Router oder Apps (z.B. Plex) die akzeptieren nur eine .pfx (PKCS#12-Standard)! Hm, und schon wären wir gekniffen ☹ ...

## 1. alles in einem Schritt, oups ...Script

Für alle die, die die Realisierung wie von mir empfohlen auf einem PI oder Linux Gerät implementiert haben, kommt hier ein Script, das schnell angepasst ist und dann eure Domäne als Parameter nimmt und alle erforderlichen Dateien automatisch erstellt.

**8-tung!** Das Stammzertifikat muss vorhanden / bereits erstellt sein! und sich im gleichen Verzeichnis befinden wie das Script.

```
#!/bin/bash

#Required
domain=$1
commonname=$domain

#Change to your company details
country=<>
state=<>
locality=<>
organization=<>
organizationalunit=<>
email=<>

if [ -z "$domain" ]
then
    echo "Argument not present."
    echo "Usage $0 [common name]"

    exit 99
```

```
fi

echo "1. generating .key for $domain"
#generate a .key
openssl genrsa -out $domain.key 2048
echo "successful!"
echo ""

echo "2. generating .csr for $domain"
#generate a .csr
openssl req -new -key $domain.key -out $domain.csr \
    -subj
"/C=$country/ST=$state/L=$locality/O=$organization/OU=$organizationalunit/CN=$commonname/email
Address=$email"
echo "successful!"
echo ""

echo "3. generating .ext for $domain"
cat > $domain.ext <<EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = $domain

EOF
echo "successful!"
echo ""

echo "4. creating .csr for $domain"
openssl x509 -req -in $domain.csr -CA <SLD>.pem -CAkey <SLD>.key -CAcreateserial -out
$domain.crt -days 3650 -sha256 -extfile $domain.ext
echo "successful!"
echo ""

echo "5. creating .pem for $domain"
cat $domain.crt $domain.key > $domain.pem
```

```
echo "successful!"
echo ""

echo "6. creating .pfx for $domain"
openssl pkcs12 -export -in $domain.pem -out $domain.pfx
echo "successful!"
echo ""
echo "-----"
echo "----- All Set! -----"
echo "-----"
echo
```

- Hier könnt ihr das Script herunterladen: [genCert.zip](#)
- entpacken und auf eure Bedürfnisse anpassen (sucht nach <> und nach <SLD>. Dann abspeichern.
- mit dem Kommando: "sudo chmod +x <scriptname>.sh" ausführbar machen

**fertig!**

ausführen mit Kommando: "./<sriptname>.sh <subDomäne.SLD.TLD>" ☐