SSL

Be your own SSL Certificate Authority.

This tutorial is based on the domain `nextcloud.home`. So change the domain to your specific domain.

It is also important that the domain address gets redirected from your router or use AdGuardHome. This can also be set in the `/etc/hosts` file of your computer, but to reach the domain on every device, it is easier to change this directly in the router or AdGuardHome:

```
nextcloud.domain SERVER-IP
```

mkcert

mkcert is a simple tool for making locally-trusted development certificates. It requires no configuration.

Packages

```
pacman -S nss mkcert
```

Create root certificate

```
mkcert -install
```

Create certificates for your domains

```
mkcert nextcloud.home
```

Manually

Generating the private key and root certificate

```
openssl genrsa -des3 -out rootCA.key 2048
```

```
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 1825 -out
rootCA.pem
```

Change the following information as you wish. It appears when you view the certificate through your browser.

```
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

Creating CA-Signed certificates for your domains

```
openssl genrsa -out nextcloud.home-key.pem 2048
```

```
openssl req -new -key nextcloud.home-key.pem -out nextcloud.home.pem
```

```
nano nextcloud.home.ext
```

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

[alt_names]
DNS.1 = nextcloud.home
```

Script

Create the file in `nano /etc/nginx/ssl/ssl.sh`.

```
#!/bin/sh

if [ "$#" -ne 1 ]
then
  echo "Usage: Must supply a domain"
  exit 1
fi

DOMAIN=$1

openssl genrsa -out $DOMAIN-key.pem 2048
openssl req -new -key $DOMAIN-key.pem -out $DOMAIN.pem

cat > $DOMAIN.ext << EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment,
dataEncipherment
subjectAltName = @alt_names
```

```
[alt_names]
DNS.1 = $DOMAIN
EOF

openssl x509 -req -in $DOMAIN.pem -CA rootCA.pem -CAkey rootCA.key -
CAcreateserial \
-out $DOMAIN.crt -days 825 -sha256 -extfile $DOMAIN.ext
```

```
chmod +x ssl.sh
./ssl.sh nextcloud.home
```

Installing your root certificate on all the devices

You'll need to create a `rootCA.pem` file on every device and copy the content of `cat rootCA.pem` file wherever you created it in section generating_the_private_key_and_root_certificate (manually).

If you used mkcert just run this command `cat $(mkcert -CAROOT)/rootCA.pem`.

Arch Linux

```
sudo trust anchor --store rootCA.pem
```

Android

User trusted credentials

`Settings` - `Security` - `Encryption and credentials` - `Install a certificate`

Check under:

`Settings` - `Security` - `Trusted credentials` - `User`

System trusted credentials

If “User trusted credentials” is not enough and you need the certificate in the system, follow the next lines. However, this requires a rooted device:

```
hashed_name=`openssl x509 -inform PEM -subject_hash_old -in rootCA.pem |
head -1` && cp rootCA.pem $hashed_name.0
ls $hashed_name.0
```

```
adb root
adb shell mount -o rw,remount /
adb push $hashed_name.0 /system/etc/security/cacerts/
adb shell chmod 644 /system/etc/security/cacerts/$hashed_name.0
```

```
adb shell chown root:root /system/etc/security/cacerts/$hashed_name.0
adb shell reboot
```

You can also use the Magisk module MagiskTrustUserCerts (Android 13) or ConscryptTrustUserCerts (Android 14) which does the same as above.

Nginx

Check also nginx

ssl-params.conf

```
nano /etc/nginx/conf.d/ssl-params.conf
```

```
ssl_protocols TLSv1.3;
ssl_prefer_server_ciphers on;
ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";
ssl_ecdh_curve secp384r1;
ssl_session_cache shared:SSL:10m;
```

example

```
server {
        listen 80;
        listen [::]:80;
        server_name nextcloud.home;

        # enforce https
        return 301 https://$server_name:443$request_uri;
}

server {
        listen 443 ssl http2;
        listen [::]:443 ssl http2;
        server_name nextcloud.home;

        ssl_certificate /etc/nginx/ssl/nextcloud.home.pem;
        ssl_certificate_key /etc/nginx/ssl/nextcloud.home-key.pem;
        include conf.d/ssl-params.conf;
        access_log /var/log/nginx/nextcloud.home_access_log;
        error_log /var/log/nginx/nextcloud.home-error_log;

        location / {
                your things;
        }
```

```
}
```