NEXTCLOUD INSTALLATIONSANLEITUNGEN

Nextcloud Talk mit Signaling-Server

von Carsten Rieger | Aktualisiert 7. November 2020

Nextcloud und Talk

Präsentationen, Videokonferenzen, Telefonate und Chats – alles mit Ihrer selbstgehosteten Nextcloud. Zögern Sie nicht länger und zeigen Sie WhatsApp, Skype, Teams und Anderen, wie ***sichere Kommunikation einfach funktioniert***. Ein dedizierter Server, basierend auf Ubuntu 20.04 LTS (64Bit) wird sowohl aus Last-, als auch aus Performancegründen benötigt. Der Signalingserver ist über eine eigene (Sub-)Domain aus dem Internet erreichbar und sollte diese Mindestvoraussetzungen erfüllen: Ubuntu 20.04.1 LTS, 4 CPU Cores, 16GB RAM und 1GBit/s (up/down). Auch auf schwächerer Hardware lässt sich der Signaling Server betreiben, es kann allerdings zu Einbußen bzgl. der Performance und der Konferenzteilnehmeranzahl kommen.

Beginnen wir zuerst mit der Installation eines stun-/coturn-Servers, um auch Gespräche von Teilnehmern hinter NAT-Infrastrukturen verlässlich zustande kommen zu lassen. Unter Ubuntu ist dafür coturn als STUN/TURN-Server verfügbar.

```
sudo -s
```

Tragen Sie Ihre öffentliche/statische IP-Aresse und ihre Domäne in die Datei /etc/hosts ein:

nano /etc/hosts

```
...
127.0.1.1 ihresignaling.domain.de
...
<öffentliche IP-Adresse> ihresignaling.domain.de
```

Installieren Sie nun den coturn-Server:

```
apt install coturn -y
```

Passen Sie die Konfiguration an:

```
nano /etc/default/coturn
```

Entfernen Sie die Raute ‚#' am Anfang der Zeile „TURNSERVER_ENABLED=1"

```
#
# Uncomment it if you want to have the turnserver running as
# an automatic system service daemon
#
TURNSERVER_ENABLED=1
```

Speichern und schließen Sie diese Datei und erzeugen dann eine neu Konfigurationsdatei namens *turnserver.conf*:

```
mv /etc/turnserver.conf /etc/turnserver.conf.bak && nano /etc/turnserver.conf
```

Kopieren Sie alle nachfolgenden Zeilen hinein:

```
tls-listening-port=5349
fingerprint
#external-ip: optional
#external-ip=<Ihre öffentliche/statische IP-Adresse>
lt-cred-mech
use-auth-secret
static-auth-secret=12121212121212121212121212121212121212121212121212121212121212
realm=ihresignaling.domain.de
total-quota=100
bps-capacity=0
stale-nonce=600
cert=/etc/letsencrypt/rsa-certs/fullchain.pem
pkey=/etc/letsencrypt/rsa-certs/privkey.pem
cert=/etc/letsencrypt/ecc-certs/fullchain.pem
pkey=/etc/letsencrypt/ecc-certs/privkey.pem
cipher-list="TLS-CHACHA20-POLY1305-SHA256:TLS-AES-256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-
POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-
SHA512:ECDHE-RSA-AES256-GCM-SHA384"
dh-file=/etc/ssl/certs/dhparam.pem
#no-loopback-peers
no-multicast-peers
no-tlsv1
no-tlsv1_1
no-stdout-log
```

Speichern und schließen Sie diese Datei und generieren dann ein neues „Secret“:

```
openssl rand -hex 32
```

Notieren Sie sich dieses direkt für später in einer separaten Textdatei unter dem Namen *coturn-static-auth-secret*. (siehe Tabelle weiter unten). Öffnen Sie die coturn-Datei erneut

```
nano /etc/turnserver.conf
```

und ersetzen sowohl

```
static-auth-secret 12121212121212121212121212121212121212121212121212121212121212
```

mit dem zuvor neu generierten Secret, als auch den Dummy-Eintrag ihre.domain.de mit Ihrer „echten“ Domain.

Sofern noch nicht vorhanden generieren Sie nun den Diffie-Hellman Schlüssel (dhparam.pem):

```
openssl dhparam -out /etc/ssl/certs/dhparam.pem 4096
```

”

Bitte haben Sie Geduld – das Generieren kann, in Abhängigkeit von der Systemleistung, wenige Minuten bis zu einigen Stunden dauern. Erst wenn dieser Schlüssel generiert wurde kann der coturn-Server neu gestartet werden.

Öffnen Sie den Port 5349 (UDP/TCP) sowohl in Ihrer UFW (Firewall),

```
ufw allow 5349/tcp && ufw allow 5349/udp
```

als auch in ihrer Firewall bzw. ihrem Router (bspw. FritzBox):

Editieren Sie den coturn-Service wie folgt:

```
nano /usr/lib/systemd/system/coturn.service
```

um später auf die SSL-Zertifikate zugreifen zu können und somit auch diese Kommunikation zu verschlüsseln.

```
User=www-data
Group=www-data
```

Starten Sie den coturn -Server neu:

```
systemctl daemon-reload && service coturn restart
```

Für die Einrichtung der benötigten Komponenten werden unterschiedliche Keys benötigt. In Ergänzung zum zu Beginn generierten coturn-Static-auth-key generieren wir noch vier weitere Keys:

SCHLÜSSEL	BEFEHL ZUM GENERIEREN
coturn-static-auth-secret (siehe oben)	openssl rand -hex 32
Janus-API-Key	openssl rand -base64 16
Hash-Key	openssl rand -hex 16
Block-Key	openssl rand -hex 16
Nextcloud-Secret-Key	openssl rand -hex 16

Wir fahren mit der Installation des aktuellen Pakets libSRTP2 fort.

```
cd /usr/local/src
wget http://de.archive.ubuntu.com/ubuntu/pool/universe/libs/libsrtp2
/libsrtp2-1_2.3.0-4_amd64.deb
apt install /tmp/libsrtp2-1_2.3.0-4_amd64.deb
rm -f libsrtp2-1_2.3.0-4_amd64.deb
```

Um den Sgnaling-Server installieren zu können müssen wir diesen selbst Kompilieren:

```
apt install -y golang-go
cd /etc/
git clone https://github.com/strukturag/nextcloud-spreed-signaling.git
cd /etc/nextcloud-spreed-signaling
make build
```

Nach wenigen Minuten ist der *build* abgeschlossen und die Konfiguration kann angepasst werden:

```
cd /etc/nextcloud-spreed-signaling
cp server.conf.in server.conf
nano server.conf
```

Bearbeiten Sie die folgenden Bereiche:

```
[http]
# IP and port to listen on for HTTP requests.
# Comment line to disable the listener.
listen = 127.0.0.1:8080
...
[sessions]
# Secret value used to generate checksums of sessions. This should be a random
# string of 32 or 64 bytes.
hashkey = 123...xyz
# hashkey = Ihr zuvor selbst generierter <Hash-Key>
...
# Optional key for encrypting data in the sessions. Must be either 16, 24 or
# 32 bytes.
# If no key is specified, data will not be encrypted (not recommended).
blockkey = abc...789
# blockkey = Ihr zuvor selbst generierter <Block-Key>
...
[backend]
# Comma-separated list of backend ids from which clients are allowed to connect
# from. Each backend will have isolated rooms, i.e. clients connecting to room
# "abc12345" on backend 1 will be in a different room than clients connected to
# a room with the same name on backend 2. Also sessions connected from different
# backends will not be able to communicate with each other.
backends = backend1
...
# Common shared secret for requests from and to the backend servers if
# "allowall" is enabled. This must be the same value as configured in the
# Nextcloud admin ui.
secret = 456...z9y
# secret = Ihr zuvor selbst generierter <Nextcloud-Secret-Key>
...
[nats]
# Url of NATS backend to use. This can also be a list of URLs to connect to
# multiple backends. For local development, this can be set to ":loopback:"
# to process NATS messages internally instead of sending them through an
# external NATS backend.
url = nats://localhost:4222

[mcu]
# The type of the MCU to use. Currently only "janus" and "proxy" are supported.
# Leave empty to disable MCU functionality.
#type =

# For type "janus": the URL to the websocket endpoint of the MCU server.
# For type "proxy": a space-separated list of proxy URLs to connect to.
#url =
url = ws://127.0.0.1:8188
...
[turn]
```

```
# API key that the MCU will need to send when requesting TURN credentials.
apikey = Zyx...+#=
# secret = Ihr zuvor selbst generierter <Janus-API-Key>

# The shared secret to use for generating TURN credentials. This must be the
# same as on the TURN server.
secret = 1212...1212
# secret = Ihr zu Beginn selbst generierters coturn-static-auth-secret

# A comma-separated list of TURN servers to use. Leave empty to disable the
# TURN REST API.
#servers = turn:1.2.3.4:9991?transport=udp,turn:1.2.3.4:9991?transport=tcp
servers =
turn:ihresignaling.domain.de:5349?transport=udp,turn:ihresignaling.domain.de:5349?transport
...
```

Es folgt die Janus Installation

```
apt install -y janus
```

gefolgt von der Konfiguration

```
nano /etc/janus/janus.jcfg
```

Passen Sie die Konfiguration wie folgt an:

```
nat: {
stun_server = "ihresignaling.domain.de"
stun_port = 5349
...
full_trickle = true
...
turn_rest_api_key = "Zyx...+#="
# turn_rest_api_key = Ihr zuvor selbst generierter <Janus-API-Key>
...
```

Janus wird nur lokal angesprochen und daher wie folgt konfiguriert:

```
nano /etc/janus/janus.transport.http.jcfg
```

```
...
interface = "lo"
...
```

Ebenfalls für den Websocket

```
nano /etc/janus/janus.transport.websockets.jcfg
```

```
...
ws_interface = "lo"
...
```

Um NATS verwenden zu können, bereiten wir das System noch für Docker-CE vor:

```
apt remove docker docker-engine docker.io
apt install -y apt-transport-https ca-certificates curl software-properties-common
```

```
echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable" | tee
/etc/apt/sources.list.d/docker.list
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
apt update && apt install -y docker-ce
```

Laden Sie nun NATS als Docker herunter:

```
docker pull nats:latest
```

```
docker run --name=NATSSERVER -d -p 4222:4222 -ti --restart=always nats:latest
```

Richten wir nun den Webserver ein.

```
echo "deb http://nginx.org/packages/mainline/ubuntu $(lsb_release -cs) nginx" | tee
/etc/apt/sources.list.d/nginx.list
```

```
curl -fsSL https://nginx.org/keys/nginx_signing.key | apt-key add -
```

```
apt update && apt install -y nginx
```

```
systemctl enable nginx.service
```

```
mv /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak && touch /etc/nginx/nginx.conf
```

```
nano /etc/nginx/nginx.conf
```

Kopieren Sie den gesamten nachfolgenden Inhalt in die Datei:

```
user www-data;
worker_processes auto;
pid /var/run/nginx.pid;
events {
worker_connections 1024;
multi_accept on; use epoll;
}
http {
server_names_hash_bucket_size 64;
access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log warn;
set_real_ip_from 127.0.0.1;
real_ip_header X-Forwarded-For;
real_ip_recursive on;
include /etc/nginx/mime.types;
default_type application/octet-stream;
sendfile on;
send_timeout 3600;
tcp_nopush on;
tcp_nodelay on;
open_file_cache max=500 inactive=10m;
open_file_cache_errors on;
keepalive_timeout 65;
reset_timedout_connection on;
server_tokens off;
resolver 127.0.0.53 valid=30s;
resolver_timeout 5s;
include /etc/nginx/conf.d/*.conf;
}
```

und legen notwendige Verzeichnisse an:

```
mkdir -p /var/www/letsencrypt/.well-known/acme-challenge /etc/letsencrypt/rsa-certs
/etc/letsencrypt/ecc-certs
```

Erstellen Sie die notwendigen vHost-Dateien an:

```
[ -f /etc/nginx/conf.d/default.conf ] && mv /etc/nginx/conf.d/default.conf /etc/nginx
/conf.d/default.conf.bak
```

```
touch /etc/nginx/conf.d/default.conf
touch /etc/nginx/conf.d/http.conf
touch /etc/nginx/conf.d/ihresignaling.domain.de.conf
```

```
nano /etc/nginx/conf.d/http.conf
```

Kopieren Sie alle nachfolgenden Zeilen in die Datei *http.conf* und passen die rotmarkierten Werte entsprechend Ihres Systems an:

```
server {
listen 80 default_server;
listen [::]:80 default_server;
server_name ihresignaling.domain.de;
root /var/www;
location ^~ /.well-known/acme-challenge {
default_type text/plain;
root /var/www/letsencrypt;
}
location / {
return 301 https://$host$request_uri;
}
}
```

Wechseln Sie in die signaling-vHost-Datei:

```
nano /etc/nginx/conf.d/ihresignaling.domain.de.conf
```

und kopieren den nachfolgenden Text in diese Datei. Passen Sie die roten Werte an.

```
upstream signaling {
server 127.0.0.1:8080;
}
server {
listen 8443 ssl http2;
listen [::]:8443 ssl http2;
server_name ihresignaling.domain.de;
ssl_certificate /etc/ssl/certs/ssl-cert-snakeoil.pem;
ssl_certificate_key /etc/ssl/private/ssl-cert-snakeoil.key;
ssl_trusted_certificate /etc/ssl/certs/ssl-cert-snakeoil.pem;
#ssl_certificate /etc/letsencrypt/rsa-certs/fullchain.pem;
#ssl_certificate_key /etc/letsencrypt/rsa-certs/privkey.pem;
#ssl_certificate /etc/letsencrypt/ecc-certs/fullchain.pem;
#ssl_certificate_key /etc/letsencrypt/ecc-certs/privkey.pem;
#ssl_trusted_certificate /etc/letsencrypt/ecc-certs/chain.pem;
ssl_dhparam /etc/ssl/certs/dhparam.pem;
ssl_session_timeout 1d;
ssl_session_cache shared:SSL:50m;
ssl_session_tickets off;
ssl_protocols TLSv1.3 TLSv1.2;
ssl_ciphers 'TLS-CHACHA20-POLY1305-SHA256:TLS-AES-256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-
SHA384';
ssl_ecdh_curve X448:secp521r1:secp384r1;
ssl_prefer_server_ciphers on;
ssl_stapling on;
ssl_stapling_verify on;
location /standalone-signaling/ {
proxy_pass http://signaling/;
proxy_http_version 1.1;
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
location /standalone-signaling/spreed {
proxy_pass http://signaling/spreed;
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "Upgrade";
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
}
```

```
service nginx restart
```

Bitte stellen Sie sicher, dass Ihr Server sowohl über Port 80/TCP als auch über Port 443/TCP von außen erreichbar ist. Das Erstellen und Aktualisieren von Let's Encryptzertifikaten erfolgt zwingend über http und Port 80! Für das Zertifikatshandling erstellen wir nun einen dedizierten Benutzer und fügen diesen der www-data Gruppe hinzu:

```
adduser --disabled-login acmeuser
```

```
usermod -a -G www-data acmeuser
```

Wechseln Sie in die Shell des neuen Benutzers (acmeuser) um die Zertifikatssoftware zu installieren und verlassen diese Shell danach wieder:

```
su - acmeuser
```

```
curl https://get.acme.sh | sh
```

```
exit
```

Passen Sie die entsprechenden Berechtigungen an, um die neuen Zertifikate darin speichern zu können:

```
chmod -R 775 /var/www/letsencrypt /etc/letsencrypt && chown -R www-data:www-data
/var/www/ /etc/letsencrypt
```

Wechseln Sie erneut in die Shell des neuen Benutzers

```
su - acmeuser
```

und requestieren (beantragen) die SSL-Zertifikate. Ersetzen Sie dabei **ihre.domain.de** mit Ihrer Domain :

```
acme.sh --issue -d ihresignaling.domain.de --keylength 4096 -w /var/www/letsencrypt
--key-file /etc/letsencrypt/rsa-certs/privkey.pem --ca-file /etc/letsencrypt/rsa-
certs/chain.pem --cert-file /etc/letsencrypt/rsa-certs/cert.pem --fullchain-file
/etc/letsencrypt/rsa-certs/fullchain.pem
```

```
acme.sh --issue -d ihresignaling.domain.de --keylength ec-384 -w /var/www/letsencrypt
--key-file /etc/letsencrypt/ecc-certs/privkey.pem --ca-file /etc/letsencrypt/ecc-
certs/chain.pem --cert-file /etc/letsencrypt/ecc-certs/cert.pem --fullchain-file
/etc/letsencrypt/ecc-certs/fullchain.pem
```

Verlassen Sie die Shell des neuen Benutzers

```
exit
```

Entfernen Sie Ihre bisher verwendeten Self-Signed-Zertifikate aus nginx und aktivieren Sie die neuen, vollwertigen und bereits gültigen SSL Zertifikate von Let's Encrypt. Starten Sie dann den Webserver neu:

```
sed -i '/ssl-cert-snakeoil/d' /etc/nginx/conf.d/nextcloud.conf
sed -i s/#\ssl/\ssl/g /etc/nginx/conf.d/ihresignaling.domain.de.conf
service coturn restart && service nginx restart
```

Überprüfen Sie ihren stun-/coturn-Server über diesen externen Link . Fügen Sie, wie im Beispiel dargestellt, anstatt des Beispiels „stun:ihre.domain.de:5349" ihren Domainnamen und Port ein.

Werden Ihnen Einträge vom Typ *„srflx"* und in der Spalte ‚Priority' in der letzten Zeile *„Done"* angezeigt, so funktioniert ihr coturn-Server einwandfrei! Nur in diesem Fall fahren wir mit der Installation und Einrichtung fort.

Nun starten wir den Signaling-Server. Beginnen wir mit Janus-Service:

```
service janus restart
```

Die Software wird initial manuell gestartet:

```
cd /etc/nextcloud-spreed-signaling
```

```
./bin/signaling --config server.conf
```

Startet die Anwendung fehlerfrei, so beenden wir diese (STRG+c) und richten sowohl einen technischen Benutzer, als auch einen neuen Systemdienst ein:

```
groupadd signaling
useradd --system --gid signaling --shell /usr/sbin/nologin --comment "Standalone
signaling server for Nextcloud Talk." signaling
```


Die Konfigurationsdatei des Servers wird persistent gesichert, um auch bei Updates nicht überschrieben zu werden. Des Weiteren werden die entsprechenden Berechtigungen gesetzt:

```
mkdir -p /etc/signaling
cp server.conf /etc/signaling/server.conf
chmod 600 /etc/signaling/server.conf
chown signaling: /etc/signaling/server.conf
```

Der Dienst wird nun wie folgt erstellt:

```
nano dist/init/systemd/signaling.service
```

Bitte kopieren Sie den gesamten Text hinein:

```
[Unit]
Description=Nextcloud Talk signaling server

[Service]
ExecStart=/etc/nextcloud-spreed-signaling/bin/signaling --config /etc/signaling
/server.conf
User=signaling
Group=signaling
Restart=on-failure

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

Abschließend wird der Dienst am Server registriert und gestartet:

```
cp dist/init/systemd/signaling.service /etc/systemd/system/signaling.service
systemctl enable signaling.service && systemctl start signaling.service
```

Wechseln Sie nun in Ihre Nextcloud: aktivieren und konfigurieren als Nextcloud-Administrator die Nextcloud-Talk-App wie folgt.

STUN-Server:

```
* ihresignaling.domain.de:5349
```

TURN-Server:

```
* ihresignaling.domain.de:5349
* coturn-static-auth-secret
* UDP und TCP
```

Signaling-Server:

```
* https://ihresignaling.domain.de/standalone-signaling
* Nextcloud-Secret-Key
```

Herzlichen Glückwunsch! Sie haben ihre Nextcloud um einen Signaling-Server und somit um die Nextcloud- Talk Funktionalitäten erfolgreich erweitert!

Meine Frau, meine Zwillinge und ich würden sich über eine Spende sehr freuen. Vorab vielen Dank!

AUTOR

Carsten Rieger

Carsten Rieger ist ein angestellter Senior IT-Systemengineer und zudem auch als Kleinunternehmer (Freelancer) aktiv. Er arbeitet seit 2005 im Linux- und Microsoftumfeld, ist ein Open Source Enthusiast und hoch motiviert, Linux Installationen und Troubleshooting durchzuführen. Dabei arbeitet er vorrangig mit Debian und Ubuntu Linux, Nginx und Apache Webservern, MariaDB, PHP, Cloud Infrastrukturen (bspw. Nextcloud) und auch vielen anderen Open Source Projekten (bspw. HAProxy, Jitsi, BigBlueButton, CheckMK etc.). Zudem engagiert er sich ehrenamtlich für die Dr. Michael & Angela Jacobi Stiftung - und das schon seit 2012.

32 BEITRÄGE