Get your own STUN/TURN service running

This documentation is about getting your own STUN/TURN service running, either "on premise", in your infrastructure. Or through an external specialized provider.

On premise

In the following we'll setup Coturn, a FOSS TURN server.

This can be a first step to diagnose issues then move on to a SaaS as explained in the second part. Or a permanent solution.

Requirements

To get started, you can setup a Debian 10 "buster" server with

- 1 CPU
- 1 GB RAM
- 8 GB disk mounted on /

The required Network bandwidth depends a lot on the intended usage. For STUN only, a basic connection is good enough. For TURN you'll need a good one, as much as the traffic going through the TURN server.

Latency is of utmost importance, so make sure to set up your server close to your end users.

Network security wise, with the default Coturn values you'll need the following ports

- 3478 TCP and UDP: TURN listener port
- 5349 TCP and UDP: TURN listener port for TLS and DTLS
- 3479 TCP and UDP: alternative TURN listener port
- 5350 TCP and UDP: alternative TURN listener port for TLS and DTLS
- 9641 TCP: Prometheus metrics, requires Coturn 4.5.2 yet to be released
- 9090 TCP: for the web admin UI if you want to enable it, which is very optional
- 49152 to 65535 UDP: relay endpoints range

Those value can be changed, for example you can set up the alternate ports to 80 & 443 and reduce the relay endpoints range.

Coturn setup

Shortly explained, you've to install Coturn with `apt install coturn`, configure it in `/etc/turnserver.conf`, then (re)start the service `systemctl restart coturn`.

Instead of documenting the setup here, we provide two examples in this repository:

- A bash based setup, you can follow the commands in `bash/setup.sh` to set up Coturn manually.
- An Ansible based setup. Have a look at `ansible/requirements.yml` for the required roles, `ansible/playbook.yml` for an example playbook. Have a look in Coturn role defaults for all the available variables.

Note that Coturn `/etc/turnserver.conf` configuration file is extensively commented and you can ask questions opening an issue in Github.

Enable TLS/DTLS

To enable TLS and DTLS on your Coturn server, you have to set up your own certificate on the server.

That won't be explained in details here, but you can use certbot to get a certificate from let's encrypt. Here is geerlingguy.certbot, a good Ansible role to automate that.

Once the certificate & private key, update the configuration in `/etc/turnserver.conf`.

```
# Make sure to comment those
#no-tls
#no-dtls
# Enforce safer cipher
cipher-list="ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECD
# Specify the path to your public certificate fullchain & private key
cert=/etc/letsencrypt/live/<your_fqdn>/fullchain.pem
pkey=/etc/letsencrypt/live/<your_fqdn>/privkey.pem
# Allow TLS 1.3 only
no-tlsv1
no-tlsv1_1
no-tlsv1_2
```

And `systemctl restart coturn`.

If you're using our Ansible role, change the related variables

```
coturn_tls_enabled: true
coturn_dtls_enabled: true
coturn_tls_cert: /etc/letsencrypt/live/<your_fqdn>.cloud/fullchain.pem
coturn_tls_key: /etc/letsencrypt/live/<your_fqdn>.cloud/privkey.pem
```

Authenticated TURN

The access to the TURN server is unauthenticated by default.

You can enable a simple authentification, with users defined in the configuration `/etc/turnserver.conf` file:

```
lt-cred-mech
user=turnuser:turnuserpassword
user=turnuser2:turnuserpassword2
```

And `systemctl restart coturn`

Read the configuration file's comments to know how to replace the cleartext password by a hash, or rely on a database, or oauth.

For our Ansible role, add those variables

```
coturn_users:
  - login: turnuser
    password: turnpassword
  - login: turnuser2
    password: turnpassword2
```

External providers

Here you have quickstarts about TURN SaaS.

Xirsys

Xirsys is a "Global TURN server infrastructure for powering WebRTC applications and services". They provide a global service, free STUN, and a free plan including 500 MB bandwidth for TURN.

- Create yourself an account: https://global.xirsys.net/dashboard/signup
- Once connected, in the landing page create your first "channel"
- Then go in your dashboard Services tab
- For the newly created "channel", click on the "Static TURN credentials" button, then the "+" button
- You'll then have your credentials, in a JSON like format looking like this

```
iceServers: [
    {
        urls: [
            "stun:eu-turn1.xirsys.com"
        ]
    },
    {
        username: "HwvzUONJFefuOB70FAAAAAAAAAAAAABBBBBBBBBBBBBCCCCCCCCCCCCC(
        credential: "40381e6c-0000-aaaa-1111-bbbb2222cccc",
        urls: [
            "turn:eu-turn1.xirsys.com:80?transport=udp",
            "turn:eu-turn1.xirsys.com:3478?transport=udp",
            "turn:eu-turn1.xirsys.com:80?transport=tcp",
            "turn:eu-turn1.xirsys.com:3478?transport=tcp",
            "turns:eu-turn1.xirsys.com:443?transport=tcp",
            "turns:eu-turn1.xirsys.com:5349?transport=tcp"
        ]
    }
]
```

- Then you can have to configure your Platform with those.

Note, you can check your usage in the dashboard Analytics, usage alerts are also available. And here's their own get started documentation.