STUN / TURN server

Some networks have complicated NAT in front of them which can make it a little difficult to open direct connections using Websockets (like jitsi does) or similar methods between two devices. Also, some firewalls can be very strict about what connections they allow, which can lead to connection issues in Jitsi.

For these cases, we have two solutions:
the STUN server is a public server with no NAT before it at all, it's just a publicly reachable server with a public, fixed (statically assigned) IP address. so when two clients have issues getting through NATs and firewalls because of the nature of their networks, they can both connect to the STUN server, which makes the initial communication between the two easier and can allow them to reach each other directly through some very smart methods. Here's a quick summary from a youtube commend on STUN:

> The main hole punching technique was not explained here. It works as follows:
> - A and B gather the public IP and port of each other through a third party server.
> - A and B start sending packets to each other. Let's say A went first.
> - Packet from A to B is blocked by B's NAT because it was not a response.
> - Packet from B to A was sent to A because its NAT identified it as a response to the above packet.
> - Packet from A to B is also identified as a response. Hole punching is achieved.

i have read a nicer explanation somewhere but can't find it any more.

The second solution is TURN - pretty much just routing the entire traffic through a TURN server. this costs bandwidth, so there's usually no public, open TURN servers - we set up our own just in case someone is behind really strict NAT or firewalls.

To run a STUN / TURN server, we install coturn on our vps running debian, and put the following configs in /etc/stunserver.conf

```
# summary, the actual file contains way more, but the defaults are usually good


# TURN listener port for UDP and TCP (Default: 3478).
# Note: actually, TLS & DTLS sessions can connect to the
# "plain" TCP & UDP port(s), too - if allowed by configuration.
#
```

```
listening-port=3478

# TURN listener port for TLS (Default: 5349).
# Note: actually, "plain" TCP & UDP sessions can connect to the TLS & DTLS
# port(s), too - if allowed by configuration. The TURN server
# "automatically" recognizes the type of traffic. Actually, two listening
# endpoints (the "plain" one and the "tls" one) are equivalent in terms of
# functionality; but Coturn keeps both endpoints to satisfy the RFC 5766 specs.
# For secure TCP connections, Coturn currently supports
# TLS version 1.0, 1.1 and 1.2.
# For secure UDP connections, Coturn supports DTLS version 1.
#
tls-listening-port=5349



# Listener IP address of relay server. Multiple listeners can be specified.
# If no IP(s) specified in the config file or in the command line options,
# then all IPv4 and IPv6 system IPs will be used for listening.
#
#listening-ip=172.17.19.101
#listening-ip=10.207.21.238
#listening-ip=2607:f0d0:1002:51::4
listening-ip=65.108.90.108
listening-ip=2a01:4f9:c010:d401::1
# note by lucas: these are the current public ipv4 / ipv6 addresses of the Hetzner vps.

# Uncomment to use long-term credential mechanism.
# By default no credentials mechanism is used (any user allowed).
#
lt-cred-mech

# 'Static' user accounts for the long term credentials mechanism, only.
# This option cannot be used with TURN REST API.
# 'Static' user accounts are NOT dynamically checked by the turnserver process,
# so they can NOT be changed while the turnserver is running.
#
#user=username1:key1
#user=username2:key2
# OR:
#user=username1:password1
```

```
#user=username2:password2
#
# Keys must be generated by turnadmin utility. The key value depends
# on user name, realm, and password:
#
# Example:
# $ turnadmin -k -u ninefingers -r north.gov -p youhavetoberealistic
# Output: 0xbc807ee29df3c9ffa736523fb2c4e8ee
# ('0x' in the beginning of the key is what differentiates the key from
# password. If it has 0x then it is a key, otherwise it is a password).
#
# The corresponding user account entry in the config file will be:
#
#user=ninefingers:0xbc807ee29df3c9ffa736523fb2c4e8ee
# Or, equivalently, with open clear password (less secure):
#user=ninefingers:youhavetoberealistic
#
user=shinyjitsi:<password>


# The default realm to be used for the users when no explicit
# origin/realm relationship is found in the database, or if the TURN
# server is not using any database (just the commands-line settings
# and the userdb file). Must be used with long-term credentials
# mechanism or with TURN REST API.
#
# Note: If the default realm is not specified, then realm falls back to the host domain name.
#       If the domain name string is empty, or set to '(None)', then it is initialized as an
empty string.
#
realm=vps.shiny.space


# Option to hide software version. Enhance security when used in production.
# Revealing the specific software version of the agent through the
# SOFTWARE attribute might allow them to become more vulnerable to
# attacks against software that is known to contain security holes.
# Implementers SHOULD make usage of the SOFTWARE attribute a
# configurable option (https://tools.ietf.org/html/rfc5389#section-16.1.2)
#
```

```
no-software-attribute
```

then start the service and allow the specified ports in the firewall (here ufw)

Revision #3
Created 2024-10-10 00:13:03 UTC by Admin
Updated 2025-03-06 14:56:32 UTC by Admin