Reverse SSH-tunnel - Access your home server without router configuration

Release 1.2.0

Jens Getreu

Jan 04, 2019

Contents:

Accessing your home server from outside your local network is usually done by forwarding a port of your server through the router. This note describes a different approach allowing to establish a peer-to-peer connection between hosts on different private networks without having access to the router.

Routers support several technologies to provide access from the Internet to your local network. The most common solution consists of configuring various services from within the router requiring administrator access:

1. Assign a fix local IP address to your home server via DNS.
2. Forward a port on your router to the home server's port.
3. Subscribe to a free dynamic DNS service on the Internet and configure the router to use it.

The solution suggested in this note gets along without any router configuration! All you need is an external *OpenSSH* server (hereafter referred to as "gateway") with a public IP address somewhere on the Internet.

Technically, the home server establishes a reverse SSH tunnel to the gateway server. The mobile computer (hereafter called the "laptop") connects to the gateway with an SSH tunnel. Finally both tunnels are interconnected.

The underlying technology is known as "hole punching." and is one of the most effective methods of establishing peer-to-peer communication between hosts on different private networks. [srisuresh2008] documents the theoretical aspects of hole punching for both UDP and TCP, and details the crucial aspects of both application and NAT behavior that make hole punching work.

Network topology.:

```
[HomeServer:22] running SSH-server and autoSSH-client
    ↓
[router] running NAT
    ↓
[Gateway:222(14321)] running SSH-server
    ↑
[router] running NAT
    ↑
[Laptop:24321] running SSH-client
```

1 Configure the servers

1.1 Configure the *Gateway*-server

We assume that the *Gateway's* `OpenSSH`-server listens on port 222 and that it has a static public IP or a DNS-domain-name, for example `gateway.myownserver.org`. If your SSH-server listens to its default port 22, replace 222 by 22 in the following examples. If the gateway has no static IP use some dynamic DNS service.

You only need to run the following configuration steps once.

1. Install *OpenSSH* server:

   ```
   Gateway$ sudo apt install openssh-server
   ```

2. Configure

   Add to or change in `/etc/ssh/sshd_config`:

   ```
   Gateway$ sudo nano /etc/ssh/sshd_config


   ClientAliveInterval 30
   ClientAliveCountMax 99999
   GatewayPorts yes
   AllowTcpForwarding yes
   Port 222
   ```

   Note: The prompt `$` means the command can be executed as a normal user, the prompt `#` means the command must be executed as `root`.

3. Add user:

```
Gateway$ sudo adduser sshgateway
```

4. Restart ssh server:

```
Gateway$ sudo service ssh restart
```

1.2 Test tunnel from *HomeServer* to *Gateway*

Our *HomeServer* connects to *Gateway's* ssh-server listening on port 222. As the *HomeServer* initiates the connection, it can be behind a NAT or if the local network uses dynamic IP's. Please note that most of the following commands do not need root privileges, execute them as regular user.

Binds `[HomeServer] → [Gateway:222(14321)]`

1. Install *OpenSSH* server and client:

```
HomeServer$ sudo apt install openssh-server openssh-client
```

2. Generate ssh-keys:

   Skip this step if you have a `~/.ssh/id_rsa.pub` file in you home directory.

```
HomeServer$ ssh-keygen
```

3. Open reverse tunnel (you will be asked for *sshgateway*'s password):

```
HomeServer$ ssh -p 222 -fNC -R 14321:localhost:22 sshgateway@gateway.
↪myownserver.org
```

4. Check tunnel:

```
Gateway$ ps x | grep sshgateway
20730 ?        Ss     0:00 sshd: sshgateway [priv]

Gateway$ sudo netstat -a | grep 14321
tcp    0   0 *:14321        *:*         LISTEN
tcp6   0   0 [::]:14321     [::]:*      LISTEN
```

Troubleshooting

The above requires a working connection to *Gateway* by SSH. Check:

```
Homeserver$ slogin -v -l sshgateway -p 222 gateway.myownserver.org
```

Does *Gateway*'s firewall allow connections to TCP port 222? Try:

```
Gateway$ sudo ufw disable
```

Does the *Fail2ban* or *Sshguard* intrusion prevention software prevent you from connecting? Try:

```
Gateway$ sudo systemctl stop fail2ban
Gateway$ sudo systemctl stop sshguard
```

1.3 Configure the *HomeServer*-server

Binds permanently `[HomeServer]` → `[Gateway:222(14321)]`

The following needs to be executed only once.

1. Generate ssh-keys:

   Skip this step if you have a `~/.ssh/id_rsa.pub` file in you home directory.

   ```
   HomeServer$ ssh-keygen
   ```

2. Allow connecting `Gateway` from `HomeServer` without password (you will be asked for *sshgateway*'s password once):

   ```
   HomeServer$ ssh-copy-id -i ~/.ssh/id_rsa.pub -p 222␣
   ↪sshgateway@gateway.myownserver.org
   ```

3. Make tunnel persistent

   Choose one of the two methods *systemd* (preferred) or *crond*.

   a) Start tunnel with *systemd*

   Install `autossh`:

   ```
   HomeServer$ sudo apt-get install autossh
   ```

   Create a file `/etc/systemd/system/autossh.service`:

   ```
   nano /etc/systemd/system/autossh.service
   ```

   with the following content:

   ```
   [Unit]
   Description=Reverse SSH-tunnel to gateway
   After=network-online.target ssh.service

   [Service]
   ExecStart=autossh -p 222 -fNC -R 14321:localhost:22␣
   ↪sshgateway@gateway.myownserver.org -i /home/sshgateway/.ssh/id_
   ↪rsa

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

   Enable and start service:

   ```
   HomeServer$ sudo systemctl enable autossh.service
   HomeServer$ sudo systemctl start  autossh.service
   ```

   Verify:

   ```
   HomeServer$ sudo systemctl status  autossh.service
   ```

   b) Start tunnel with *crond*

   Install `autossh`:

```
HomeServer$ sudo apt-get install autossh
```

Configure *crontab* to start *autossh* after reboot:

```
HomeServer$ crontab -e
```

Add the following line (all in one line)

crontab -e

```
@reboot  autossh -p 222 -fNC -R 14321:localhost:22␣
↪sshgateway@gateway.myownserver.org
```

Reboot:

```
HomeServer$ sudo reboot
```

2 Connect *Laptop* to *HomeServer*

2.1 Establish tunnel from *Laptop* to *Gateway*

The following is not reboot persistent and needs to be executed once for every login session before you connect to your *HomeServer*.

Laptop is behind a NAT.

Binds `[Gateway:222(14321)]` ← `[Laptop:24321]`

a. Open tunnel:

```
Laptop$ ssh -p 222 -fNL  24321:localhost:14321 sshgateway@gateway.
↪myownserver.org
```

b. Check tunnel:

```
Gateway$ sudo netstat -a | grep 14321
...
tcp6   0   0 localhost:37109  localhost:14321   ESTABLISHED
tcp6   0   0 localhost:14321  localhost:37109   ESTABLISHED


Laptop$ sudo netstat -a | grep 24321
...
tcp    0   0 localhost:24321  *:*               LISTEN
tcp6   0   0 localhost:24321  [::]:*            LISTEN
tcp6   0   0 localhost:24321  localhost:48788   ESTABLISHED
tcp6   0   0 localhost:48788  localhost:24321   ESTABLISHED
```

2.2 Connect *Laptop* to *HomeServer* through *Gateway*

Connects via `[HomeServer:22]` ← `[Laptop:24321]` ← `[SSH-client on Laptop]`

Use cases:

- Remote shell:

```
Laptop$ slogin -l <HOME_SERVER_USER> -p 24321 localhost
```

  Example:

```
Laptop$ slogin -l root -p 24321 localhost
```

- File transfer with *MidnightCommander* `mc`:

  Menu -> File -> Shell link ... -> `<HOME_SERVER_USER>@localhost:24321` -> *Ok*

  Example:

  Menu -> File -> Shell link ... -> `root@localhost:24321` -> *Ok*

- File synchronisation with `unison`:

```
Laptop$ sudo unison /home ssh://<HOME_SERVER_USER>@localhost:24321/
↪home
```

  Example:

```
Laptop$ sudo unison /home ssh://root@localhost:24321/home
```

3 Further reading

Reverse SSH Tunnel – Schritt für Schritt

Reverse SSH Tunneling

References

[srisuresh2008] P. Srisuresh, B. Ford, and D. Kegel, “State of peer-to-peer (P2P) communication across network address translators (NATs),” 2008.