

# Internet infrastructure

Prof. dr. ir. André Mariën

The secure Shell

**SSH**

# SSH

- **Secure Shell = SSH**
- Used primarily to access Linux and Unix based systems
- Gives “shell” access
- designed as a secure replacement for telnet, rlogin, rsh
- Extended
  - Sftp: a secure FTP
  - Scp: a secure “cp” program
  - Port forwarding: tunneling solution

# SSH-2

- Some features:
  - Diffie-helman key exchange
  - Message integrity protection
  - Multiple sessions over one connection
- Free server-side solution: openSSH
  - <http://www.openssh.com/>
  - Includes the server: sshd
- Free clients:
  - Most popular: putty
    - <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

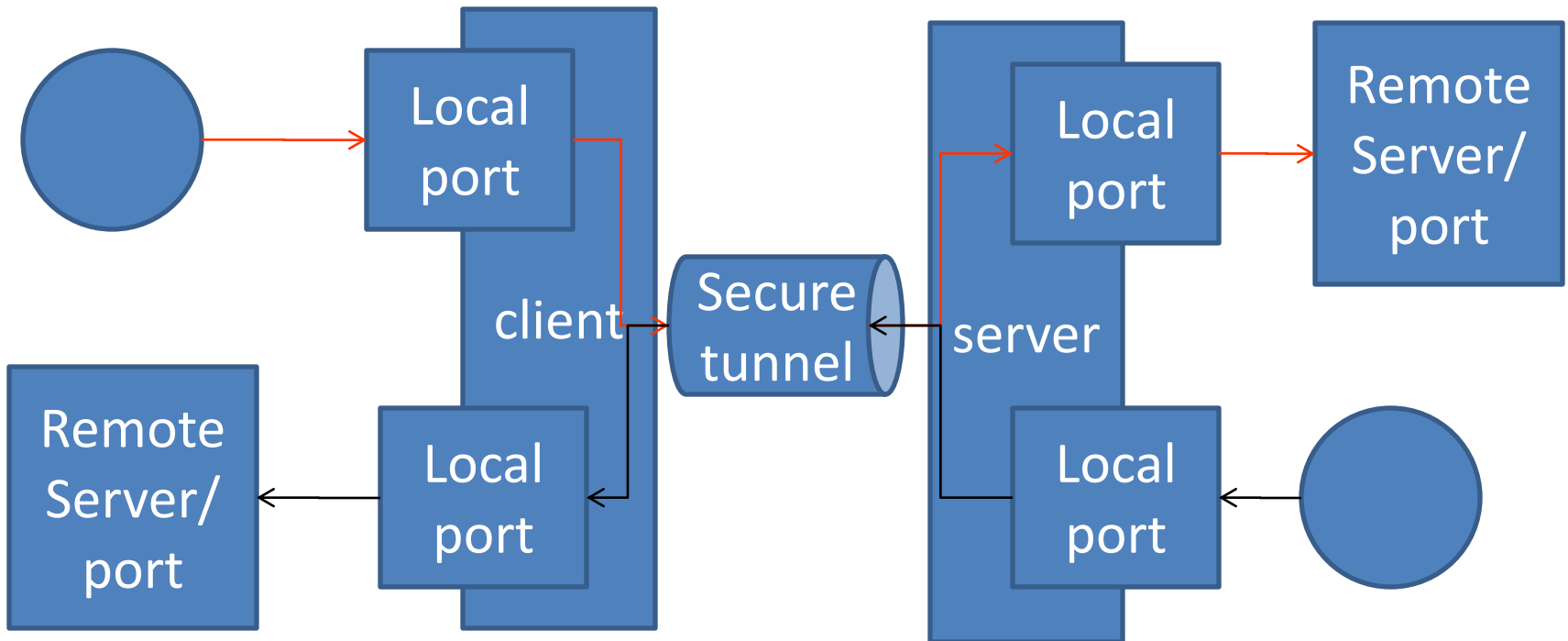
# Basic information

- Per user configuration file
  - Important to customize who can do what
- Port forwarding to server-defined locations for remote connections
  - Option: do not start a shell, just acts as port forwarder
- Authentication
  - Trusted hosts concept
  - Manual verification on fist contact, and after key changes
  - Alerts for new keys

# Making available services through the tunnel

- **-L** *[bind\_address:]port:host:hostport*
  - Start a local service on a connecting client on port “port”
  - If the user at his side connects to this local “port”, the ssh tunnel is used to set up a connection from the sshd to host:hostport server-side
  - Ssh functions as a tunnel:
    - Client entrypoint is localhost:port
    - Server connection point is host:hostport
- **-R** *[bind\_address:]port:host:hostport*
  - Reverse port mapping
  - Start a service on the sshd side, listening on port “port”
  - If connections are made, attempt to connect to “host” : “hostport” from the client

# Two tunnel uses



# SFTP & SCP

- SFTP
  - Apart from start-up, provides a command line interface much like any ordinary ftp would do
    - `sftp [user@]host[:file ...]`
    - `sftp [user@]host[:dir[/]]`
    - `sftp -b batchfile [user@]host`
- SCP
  - **Secure copy**
  - `Scp [[user@]host1:]file1 ... [[user@]host2:]file2`

# Uses

- Number one: Remote administration
  - The standard tool for remote access to systems
  - Need to have keys to access: often sufficient
  - Can use one tunnel to have remote access to the site (like from home)
  - Port forwarding can bring you to other systems
  - Port forwarding can connect you to any service on these systems
- Scheduled or automated file transfer
  - Log file recovery
  - Pushing new configurations

**STUNNEL**

# Stunnel

- “Stunnel is a program that allows you to encrypt arbitrary TCP connections inside SSL (Secure Sockets Layer) available on both Unix and Windows”
- <http://www.stunnel.org>
- “Stunnel can allow you to secure non-SSL aware daemons and protocols (like POP, IMAP, LDAP, etc) by having Stunnel provide the encryption, requiring no changes to the daemon's code.”
- Depends on an SSL library (OpenSSL, SSLeay)

# Example

- (see <http://www.stunnel.org/examples/>)
- **Forwarding an insecure port securely from one machine to another**
- POP3
  - have your machine (foo) talk to stunnel on the local machine
  - stunnel encrypts the packets and sends them another stunnel running on the remote machine (bar)
  - Remote stunnel on bar forwards them in clear text to the POP server on “bar”
- So:
  - Steps:
    - Stunnel listens on foo:pop3, forwards to bar:pop3s
    - Command:
      - `stunnel -c -d pop3 -r bar:pop3s`
    - Stunnel listens on bar:pop3s, forwards to bar:pop3.
    - Command:
      - `stunnel -p /path/to/stunnel.pem -d pop3s -r bar:pop3`

# Stunnel set-up

