SSH for Sysadmins

Vance Kochenderfer
<vance@happylemur.com>

System and Network Administrators of Pittsburgh
January 9, 2013

Things That *Won't* Be Covered

- Remote interactive logins
- Copying files with scp/sftp
- Password authentication
- Verifying host keys
- Implementations other than OpenSSH (PuTTY is a popular client for Windows that also works on Linux)
  - http://www.chiark.greenend.org.uk/~sgtatham/putty/
- Setting up VPNs
- Authentication agents (ssh-agent)

What is ssh?

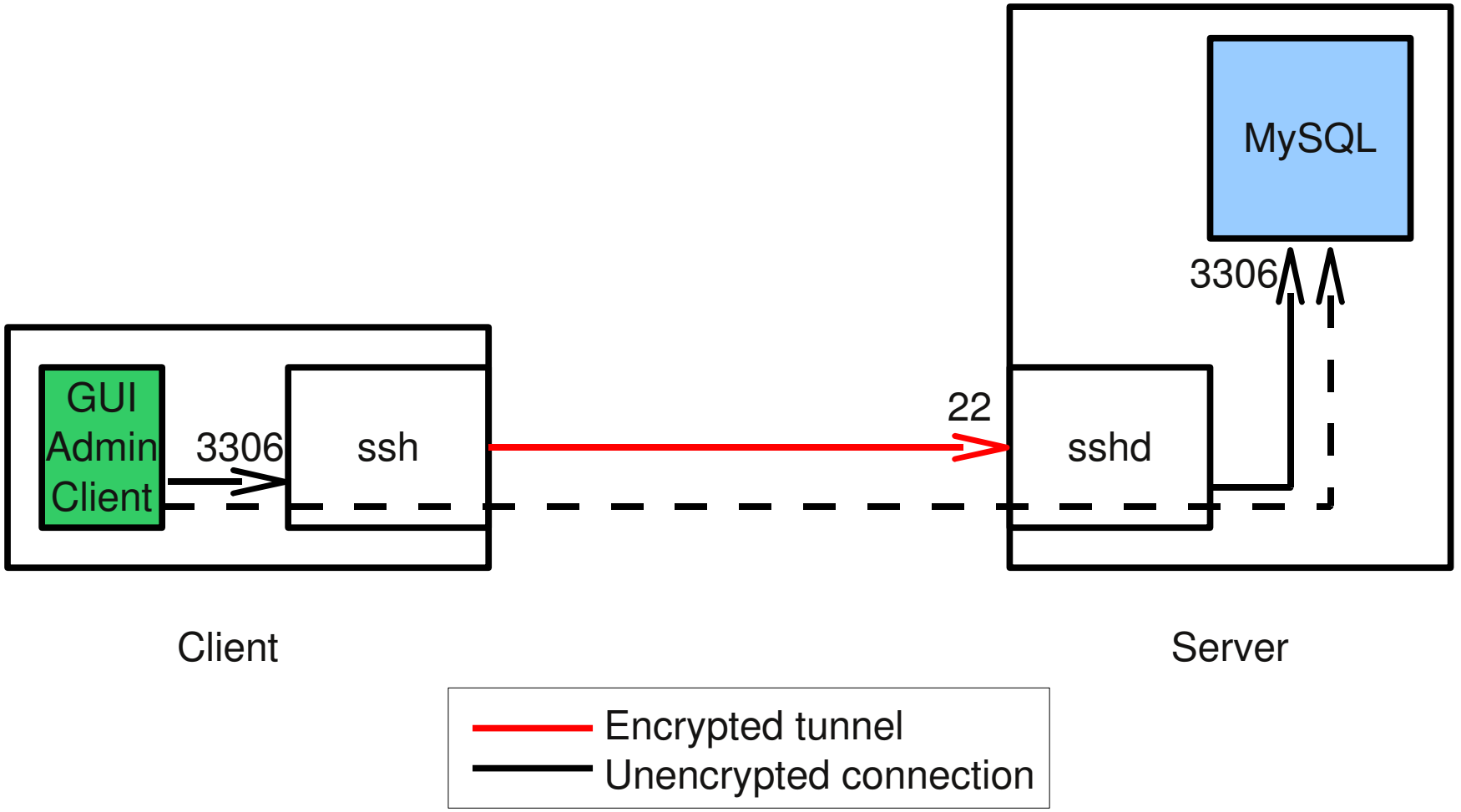
- A mechanism to establish a network connection that:
  - Authenticates the local user to the remote machine
  - Authenticates the remote machine to the local user
  - Is strongly encrypted
- ...this connection can carry arbitrary data

Tunneling: Local -> Remote

- `-L` *`[bind_addr:]port:host:host_port`*
  - `bind_addr` - local address to bind to (`localhost` [the default] for loopback only, `*` for all interfaces)
  - `port` - local port number to listen on
  - `host` - remote host to target (does not need to be the same machine receiving the SSH connection)
  - `host_port` - port number on remote host to target
- Note that only TCP (not UDP) is supported

Tunneling: Local -> Remote (2)

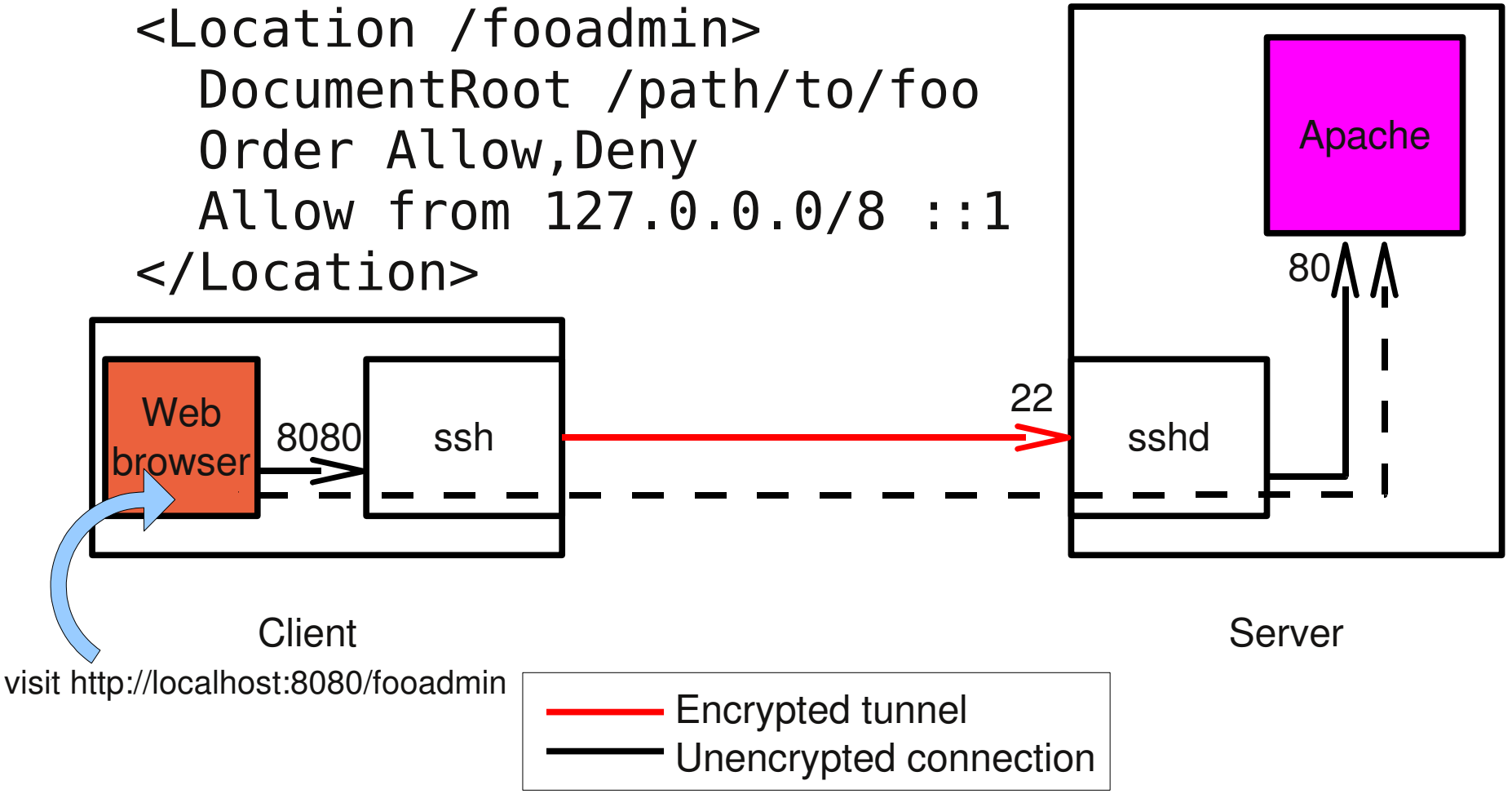
- `-L 3306:localhost:3306`



Tunneling: Local -> Remote (3)

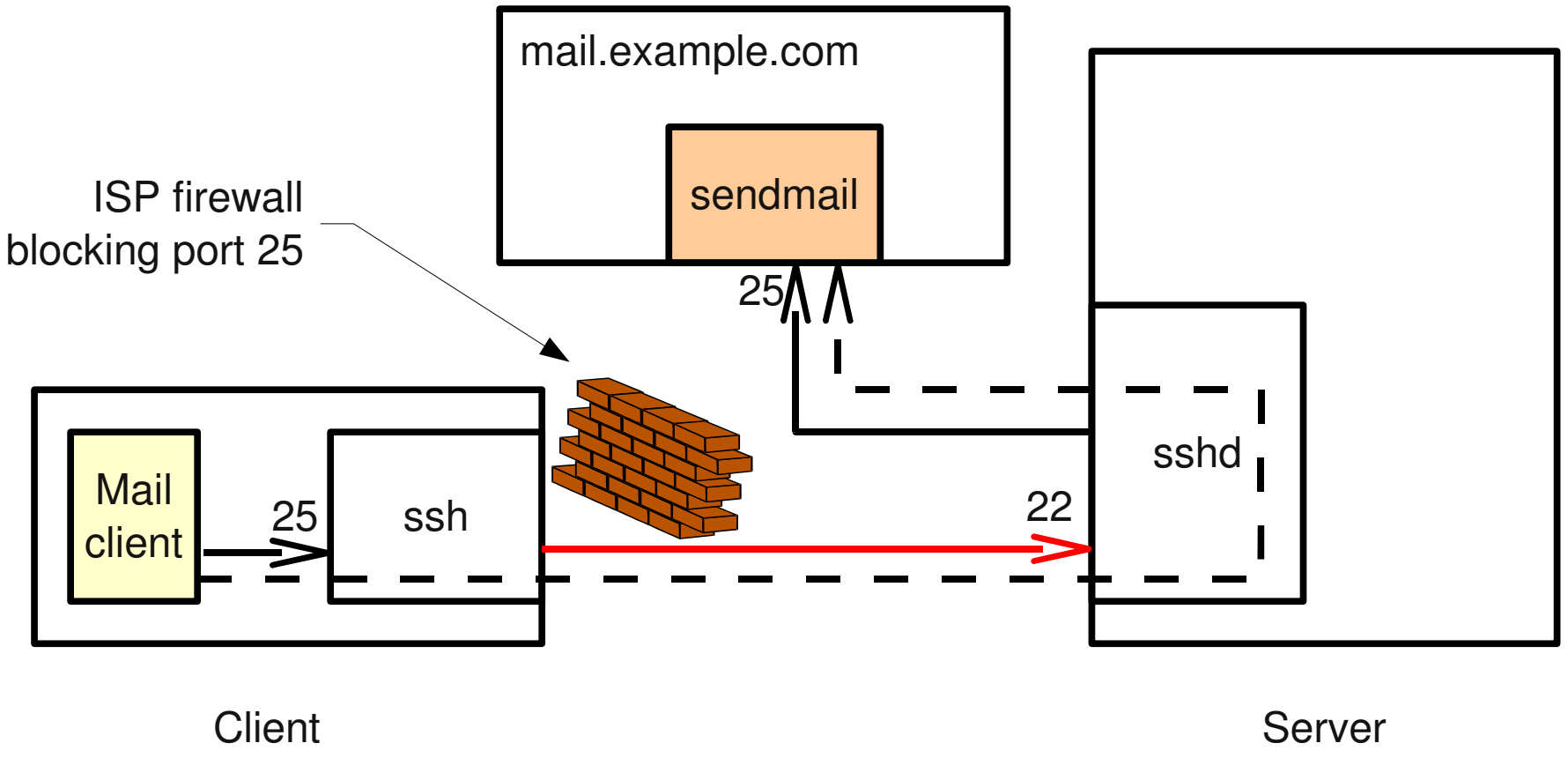
- `-L 8080:localhost:80`
- Apache config:

```
<Location /fooadmin>
  DocumentRoot /path/to/foo
  Order Allow,Deny
  Allow from 127.0.0.0/8 ::1
</Location>
```



Tunneling: Local -> Remote (4)

- `-L 25:mail.example.com:25`



Note: root-level access on client needed to bind ports < 1024

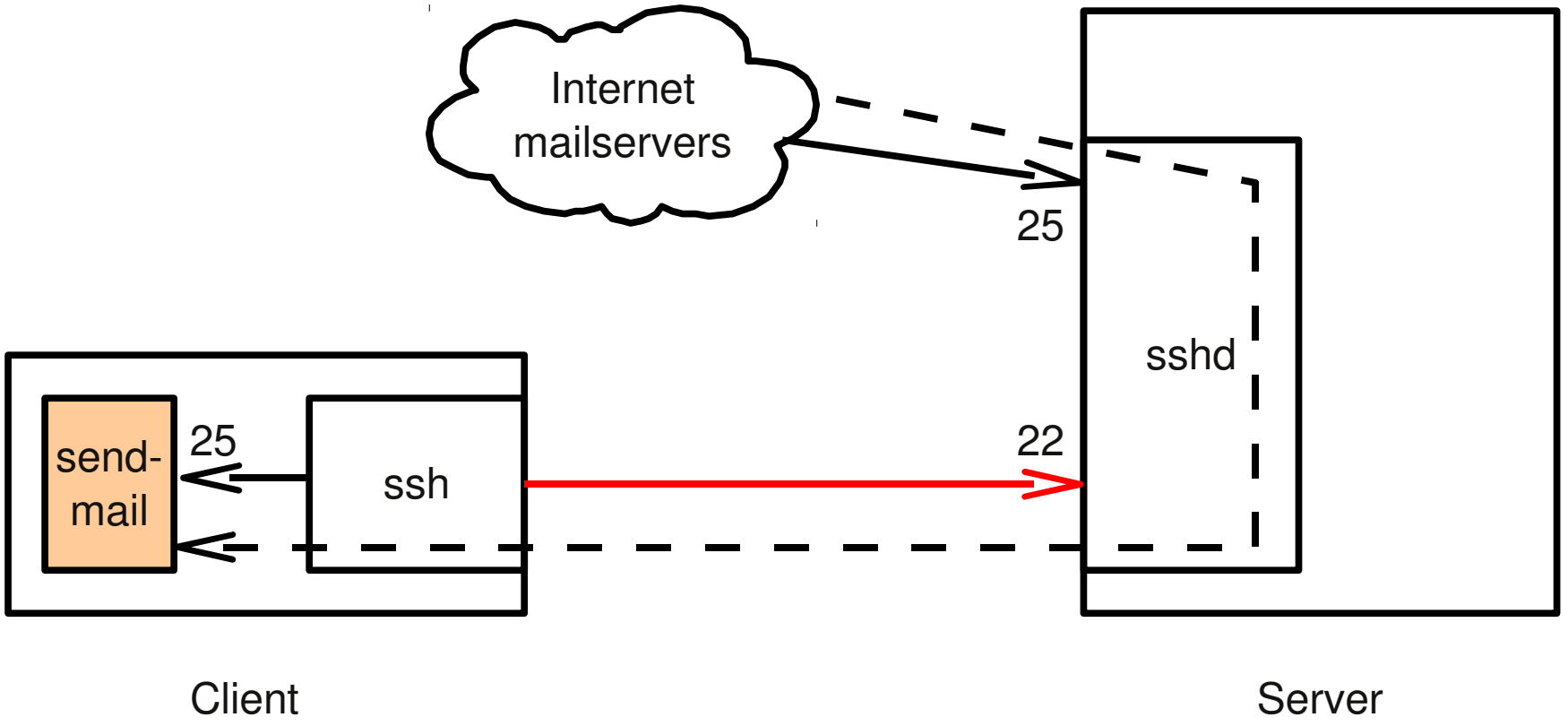
Encrypted tunnel
Unencrypted connection

Tunneling: Remote -> Local

- `-R` *`[bind_addr:]port:host:host_port`*
  - `bind_addr` - remote address to bind to (`localhost` [the default] for loopback only, `*` for all interfaces)
  - `port` - remote port number to listen on
  - `host` - host to target (does not need to be the same machine initiating the SSH connection)
  - `host_port` - port number on target host

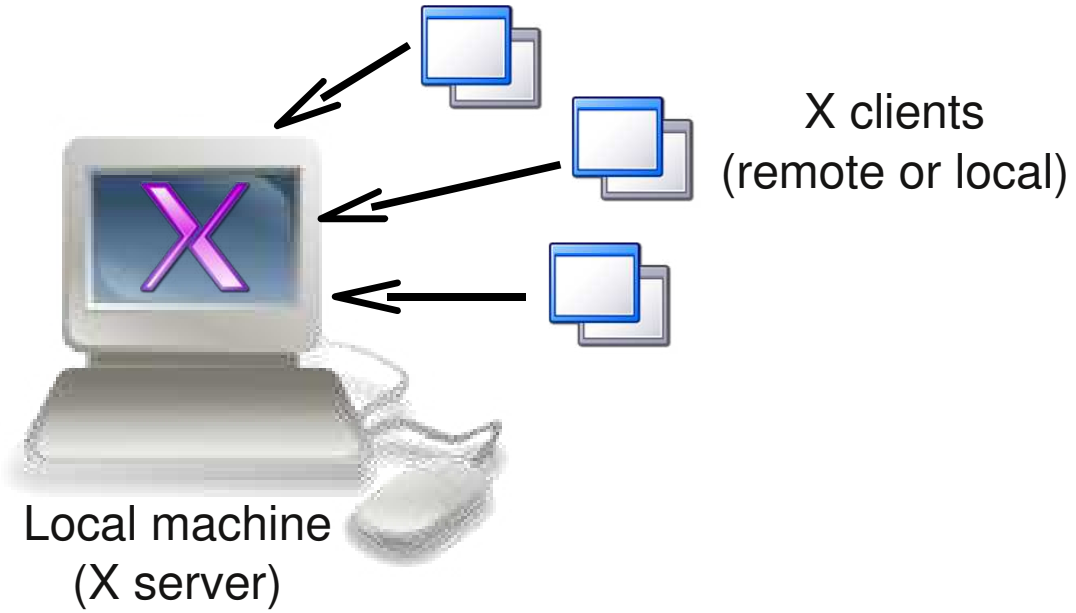
Tunneling: Remote -> Local (2)

- `-R '*:25:localhost:25'`



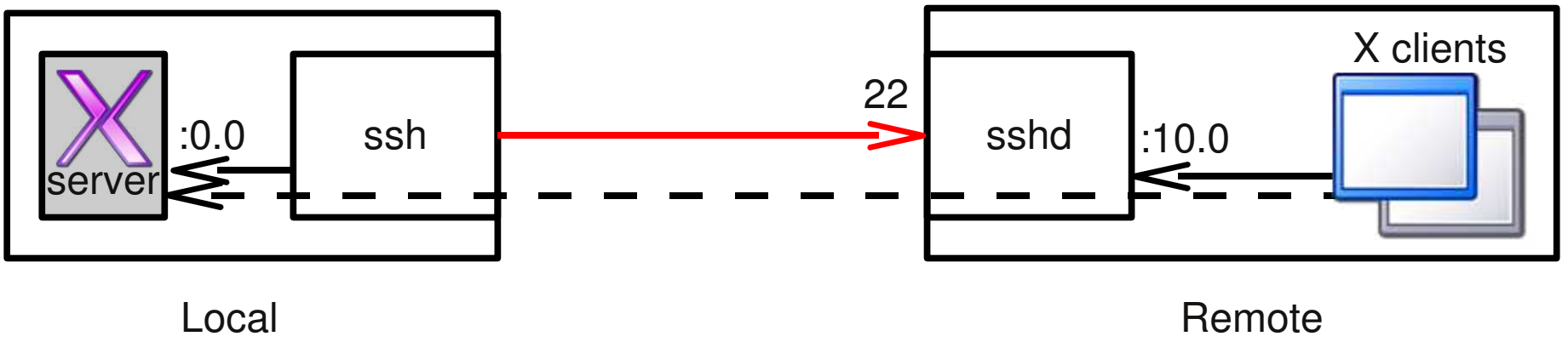
Note: root-level access on server needed to bind ports < 1024

X Forwarding



- X is an inherently network-aware protocol, but can be a pain to set up correctly and securely
- X forwarding between two machines as easy as adding `-X` to the SSH command line (or option `ForwardX11 yes`)

X Forwarding (2)



- Sets up fake X server on remote host which clients can connect to, `$DISPLAY` is auto-set
- Using compression (`-C` or `Compression yes`) is often helpful
- X protocol not very efficient over long distances; something like NX, VNC, or RDP better for frequent use

SOCKS proxy (dynamic forwarding)

- `-D` *`[bind_addr:]port`*
  - `bind_addr` - local address to bind to (`localhost` [the default] for loopback only, `*` for all interfaces)
  - `port` - local port number to listen on (1080 is IANA-assigned port for SOCKS)
- Saves having to configure port numbers
- But, applications need to support and be configured to use SOCKS

Public Key Authentication

- Symmetric vs. asymmetric ciphers
  - Symmetric (aka shared secret): sender uses a key to encrypt, receiver uses same key to decrypt
  - Asymmetric: sender uses one key (public) to encrypt, receiver uses a different key (private) to decrypt
    - Public and private keys are mathematically related, but figuring out the private key is computationally hard
    - OK for everyone to know the public key, but the private key must be protected

Public Key Authentication (2)

- Security advantages
  - With password authentication, plaintext password is made known to the remote host
    - Could be used to attack other systems where you've reused the same password
    - kernel.org compromise: http://lwn.net/Articles/464233/
  - With public key authentication, private keys are never transmitted to the remote host
  - Even if server is compromised, attacker cannot impersonate you
    - But anyone who obtains your private key and passphrase can

Public Key Authentication (3)

- Setting up
  - Generate private/public key pair: ssh-keygen
  - Set a passphrase for private key
    - Except when unattended logins are needed; in such cases, should place restriction on key
    - `-O force-command="`*`command`*`"`
    - `-O source-address=`*`address_list`*
  - Copy public key to `~/.ssh/authorized_keys` on target host (can use `ssh-copy-id` *`user@host`*)
  - OpenSSH key formats differ from other implementations; ssh-keygen and puttygen can convert between them

Host Configuration Options

- Specified in /etc/ssh/sshd_config
- `PermitRootLogin` *`value`*
  - `yes` - allow any login method (default)
  - `without-password` - don't accept password auth*
  - `forced-commands-only` - pubkey w/`-O command`
  - `no` - root cannot log in (use `su` or `sudo`)

  *This does *not* mean “public keys only” (more on this later)
- Why disable root password login?
  - Opportunistic password guessing targets root
    - 26% of attempts in http://people.clarkson.edu/~owensjp/pubs/leet08.pdf
    - 50%+ of attempts on WPLUG server
  - No other account gets even 5% of attempts
    - Protect servers using fail2ban or denyhosts

Host Configuration Options (2)

- `Port` *`number`* - port to listen on (default 22)
  - Not really a security measure
- `ListenAddr` *`host`*`|`*`IP address[:port]`*`|`*`:port`* (default all local addresses)
- `Match` *`User`*`|`*`Group`*`|`*`Host`*`|`*`Address value[,value...]`*
  - Can set custom options when the specified conditions are met

Host Configuration Options (3)

- Example: allow root to only log in from certain hosts and only with public key

```
PermitRootLogin yes
Match Address !10.0.0.0/8
   PermitRootLogin no
Match User root
   Protocol 2
   GSSAPIAuthentication no
   HostbasedAuthentication no
   ChallengeResponseAuthentication no
   PasswordAuthentication no
```

Client Configuration Options

- Specified on command line with -o (e.g., `-o "Compression no"`), ~/.ssh/config, /etc/ssh/ssh_config
  - Behavior is controlled by the *first* specified value
- `Protocol`, `*Authentication`, `Port`, `Ciphers` same as host options
  - Except that when multiple values are specified, they are tried in order (e.g., `Protocol 2,1` is different from `Protocol 1,2`)

Client Configuration Options (2)

- `ControlMaster` *`yes`*|*`no`*|*`ask`*|*`auto`*|*`autoask`*
  - Allows multiple ssh sessions to the same host to share a single connection
  - Also specify `ControlPath` *`pathname`*
    - e.g., `ControlPath ~/.ssh/master-%r@%h:%p`)
  - http://protempore.net/~calvins/howto/ssh-connection-sharing/

Client Configuration Options (3)

- `Host` *`pattern`*
  - Restricts following options (until another `Host` line is given) to hosts specified on command line matching pattern
  - Useful for making shortcuts to frequently-used hosts
  - If generic options desired, put a `Host *` line at end of config file followed by option specifications (remember, first value set for an option wins)

Client Configuration Options (4)

- Example: three hosts, plus generic options

```
Host dbserver
    HostName db.example.com
    User vkochend
    LocalForward 3306 localhost:3306
    Compression no
Host personal
    HostName somewhere.net
    User vance
    IdentityFile ~/.ssh/home-id_rsa
    ForwardX11 yes
Host secserver
    HostName auth.example.com
    Port 842
    User root
    IdentityFile ~/.ssh/work-id_rsa
    StrictHostKeyChecking yes
Host *
    Compression yes
```

Escape Character

- Gives access to some commands while connected
- Default ~, can be changed with `EscapeChar` *`char`* or disabled with `EscapeChar none` (or `-e`)
- **Only** treated specially immediately after a newline
- Some available commands
  - Disconnect (.)
  - Suspend ssh in background (Ctrl-Z)
  - Send escape character to remote system (~)
  - List available commands (?)