

# Debian/GNU Linux Remote Services

Secure Shell, Virtual Network Computing , Remote Desktops

Károly Erdei

November 6, 2014



# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling
- 5 SSH no password
- 6 VNC
- 7 X2Go
- 8 RDP

# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling
- 5 SSH no password
- 6 VNC
- 7 X2Go
- 8 RDP

# Remote Login Services

Application services to use remote hosts interactively

## Scenario: remote host offers interesting services:

- Resources (CPU, memory, disk) provided by remote host
  - Compute servers: gonzales, roadrunner, popeye, qftquad1,2,3,4
- Files located on remote host
  - **file server** for home directories
  - **scratch server** for other files
- Programs installed on remote host
  - Mathematica, Maple on compute servers

## Goal: use these remote services from local host

- Use local host as a terminal to login to remote host
- Run programs/commands on remote host
- See output on local host
  - Ascii terminal output
  - graphical output by X clients
  - some other way: vnc, x2go, rdp

# Remote Login Services

Protocols, systems

## Relevant protocols/systems:

- telnet/rlogin/rsh outdated !!
- SSH suite:
  - ssh - secure shell
  - slogin - secure login
  - sftp - secure ftp
- X-Windows X11
  - network-transparent GUI
  - too slow, not effective
- VNC - virtual network computing/console
- x2go (server, client)
- MS Windows Terminal Server
  - RDP - remote desktop protocol

# The Remote Login Server - an application program

example : the SSH Server

## Process

- **Master server** on remote host waits for new connection requests (SSH: port 22)
- For each connection, it spawns a **slave server** to handle the connection
- Multiple sessions (from the same or different clients) may be active at the same time
- Slave server handles the connection
  - gets data from local keyboard on remote host and outputs data from remote host on the local display in interaction with the local client

# Telnet is outdated

SSH is the successor

## TELNET and Rsh/Rlogin outdated - because of security problems

- All data are transferred in clear text
- Any listener between client and remote server can read everything
  - True for any unencrypted connection, think on http !!!
- telnet-ssl replaces telnet/rlogin

## Replacement: Secure Shell (ssh, slogin)

- SSH suite is the modern replacement of TELNET and rlogin
- standard protocols for secure remote access over IP networks (RFCs: 4251-5254)
- All data are **encrypted** before they are transferred via IP
- Free implementations: [www.openssh.org](http://www.openssh.org), [www.putty.org](http://www.putty.org), [www.winscp.net](http://www.winscp.net), etc.
- Commercial implementations: [www.ssh.com](http://www.ssh.com) (MS Windows)

# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling
- 5 SSH no password
- 6 VNC
- 7 X2Go
- 8 RDP



# File Services

File transfer, File sharing

## Application services to access files on remote hosts

- File transfer
  - Files are copied from one host to another
  - Command line tools: `sftp` (secure ftp), `scp` (secure remote copy), as part of the `openssh-client` package
  - Graphical tools: `gftp`, (`kasablanca`, etc.) **FileZilla** (multiplatform program)
- File sharing
  - Files are accessed from a central server
  - Files are stored and backuped on central file server
  - Client applications operate on remote files like on local files
  - Transparent file access is provided by network file systems
- Implementation:
  - NFS (Network File System) for Linux, Unix OS
  - SMB (Server Message Blocks) for MS Windows

# NFS (Network File System)

## NFS: access to remote files

- Developed by Sun Microsystems
- Used in many Intranets to interconnect file systems
- Mainly for Unix/Linux computers
- Remote file system can be accessed like local files
  - A remote file system is **mounted** to an empty local directory
  - Files below this directory can be used like local files
  - No special file transfer commands needed, no file duplication arises
- Implemented on top of UDP

## For security reasons, only used within an administrative domain

- administrativ domain:
  - an organisational unit managed by local system managers (e.g.: RISC)
  - only these managers have root access for the computer system

# FTP with gftp

Always set the protocol to SSH2

The screenshot shows the gFTP 2.0.18 application window. At the top, the menu bar includes FTP, Local, Remote, Bookmarks, Transfers, Logging, Tools, and Help. Below the menu, the connection settings are displayed: Host: hades.risc.uni-linz.ac.at, Port: (empty), User: ke, Pass: [masked], and Protocol: SSH2. The main area is split into two panes. The left pane shows the local file system at /home/ke, listing files like .AbiSuite, .adobe, .amsn, .aptitude, and kikkla. The right pane shows the remote file system at hades.risc.uni-linz.ac.at [SSH2] [All Files]\*, listing files like .AbiSuite, .acrobat, .adobe, .config, and .dbye. Below the panes is a progress bar with columns for Filename and Progress. At the bottom is a terminal window showing the following output:

```

Opening SSH connection to hades.risc.uni-linz.ac.at
Running program /usr/bin/ssh -e none -l ke -p 22 hades.risc.uni-linz.ac.at -s sftp
3: Protocol Initialization
3: Protocol version 3
Successfully logged into SSH server hades.risc.uni-linz.ac.at
1: Realpath .
1: Filenames (1 entries)
Loading directory listing /home/ke from server (LC_TIME=en_US.UTF-8)
Retrieving directory listing...
2: Open Directory /home/ke
2: File handle
3: Read Directory
3: Filenames (100 entries)
  
```

# FTP with Filezilla

Always set the port to 22

The screenshot shows the FileZilla interface with the following details:

- Host:** sftp://narwal.risc.jku.at
- Username:** ke
- Password:** [masked]
- Port:** 22
- Quickconnect:** [checked]

**Terminal Output:**

```

Response: Current directory is: "/home/ke"
Command: ls
Status: Listing directory /home/ke
Status: Calculating timezone offset of server...
Command: mtime ".x2go"
Response: 1413375064
Status: Timezone offsets: Server: 7200 seconds. Local: 3600 seconds. Difference: -3600 seconds.
Status: Directory listing successful
  
```

**Local site (/):**

Filename	Filesize	Filetype	Last modified
..pulse		Directory	10/31/2014 1...
bin		Directory	09/26/2014 1...
boot		Directory	10/31/2014 1...
data		Directory	08/07/2014 0...
data-10G		Directory	09/20/2014 0...
dev		Directory	11/05/2014 0...
etc		Directory	11/06/2014 0...
home		Directory	09/21/2014 0...
lib		Directory	10/18/2014 0...
lost+found		Directory	03/15/2014 0...
media		Directory	11/05/2014 0...
mnt		Directory	06/04/2013 0...

3 files and 24 directories. Total size: 12,852,937 bytes

**Remote site (/home/ke):**

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Grc
..					
..AbiS...		Directory	01/31/2012	drwx-----	ke kerdei
..DCO...		Directory	06/06/2011	lrwxrwx...	ke risc
..DCO...		Directory	03/26/2014	lrwxrwx...	ke risc
..DCO...		Directory	06/06/2011	lrwxrwx...	ke risc
..DCO...		Directory	06/11/2012	lrwxrwx...	ke risc
..DCO...		Directory	06/06/2011	lrwxrwx...	ke risc
..DCO...		Directory	06/06/2011	lrwxrwx...	ke risc
..Math...		Directory	10/10/201...	drwxr-xr...	ke kerdei
..Skype		Directory	06/05/2007	drwx-----	ke kerdei
..acro...		Directory	02/12/2003	drwx-----	ke kerdei
..adobe		Directory	09/26/2009	drwx-----	ke kerdei

226 files and 107 directories. Total size: 59,004,666 bytes

# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling
- 5 SSH no password
- 6 VNC
- 7 X2Go
- 8 RDP

# SSH features

## The SSH suite

### SSH - a client-server solution for network security

- client-server solution for network security
  - encryption: all data will be encrypted before sending from localhost to remote computer and vice versa
  - transparent for the user (does not notice background activities)
  - client side: login; authentication; data transfer, command execution

### SSH features

- it is a protocol: describes how to conduct secure communication over a network
- full, secure replacement for FTP and Telnet and the UNIX r-series of commands: rlogin, rsh, rcp, rexec
  - creates a secure channel for running a shell on the remote computer
  - sftp, scp is integrated in the protocol
- supports more authentication methods: password, public key, certificate, smart card, PAM and SecurID

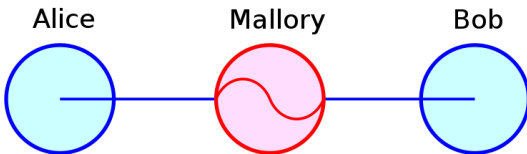
# SSH features

## Security

- uses multiple high security algorithms and strong authentication methods
  - prevents such security threats as identity spoofing and man-in-the-middle attacks
  - man-in-the-middle attack: changing the IP in the packet you communicate with the remote computer, stating: I'm the remote computer
  - man-in-the-middle attack details: next slide
- Transparent and automatic tunneling of X11 connections
- Port forwarding or SSH tunneling: for arbitrary TCP/IP-based applications, such as e-mail
- works as a proxy server, too: by the SOCKSv5 implementation
- Multiple channels that allow to have multiple terminal windows and file transfers going through one secure and authenticated connection

# SSH security

Man-in-the-middle attack - what it is





# SSH security

## Man-in-the-middle attack - conversation - I.

1. Alice sends a message to Bob, which is intercepted by Mallory:

```
Alice "Hi Bob, it's Alice. Give me your key"--> Mallory Bob
```

2. Mallory relays this message to Bob; Bob can't tell it isn't really from Alice

```
Alice Mallory "Hi Bob, it's Alice. Give me your key"--> Bob
```

3. Bob responds with his encryption key:

```
Alice Mallory <--[Bob's_key] Bob
```

4. Mallory replaces Bob's key with her own, and relays this to Alice, claiming that it is Bob's key:

```
Alice <--[Mallory's_key] Mallory Bob
```

## SSH security

### Man-in-the-middle attack - conversation II.

5. Alice encrypts a message with what she believes to be Bob's key, thinking that only Bob can read it:

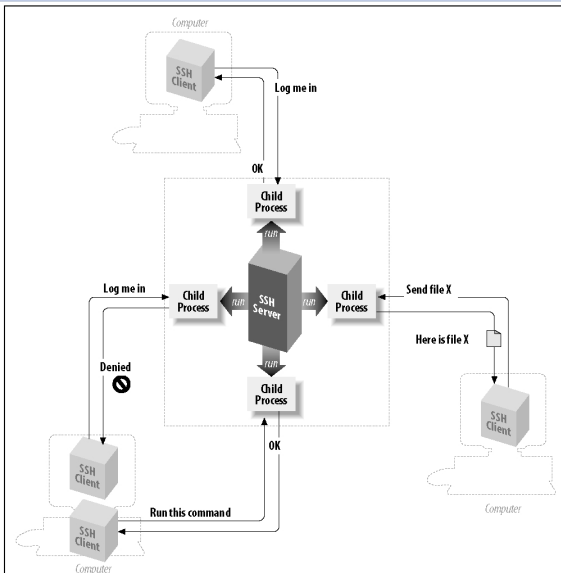
```
Alice "Meet me at the bus stop!"[encrypted with Mallory's key]
-->  Mallory      Bob
```

6. However, because it was actually encrypted with Mallory's key, Mallory can decrypt it, read it, modify it (if desired), re-encrypt with Bob's key, and forward it to Bob:

```
Alice  Mallory "Meet me at 22nd Ave!"[encrypted with Bob's key]
-->  Bob
```

7. Bob thinks that this message is a secure communication from Alice.

# The base services of SSH



# Complete Structure of the SSH protocol

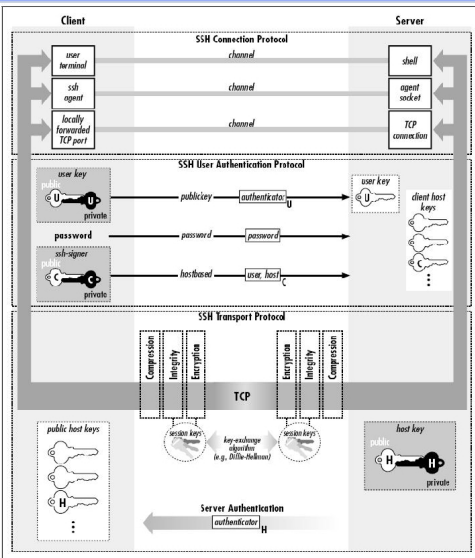


Figure 3-4 SSH-2 architecture

# The structure of the SSH-2 Protocol

## Very clean 3-layer internal architecture (RFC 4251)

- Transport Layer (RFC 4253)
  - initial key exchange, server authentication, data confidentiality, data integrity, compression, key re-exchange ( algorithm negotiation, session-ID, privacy)
- User Authentication Layer (RFC 4252)
  - Client Authentication: provides various authentication methods (public key, host bases, password, etc.)
- Connection Layer (RFC 4254)
  - defines the logical channels and the requests to handle the services like: secure interactive shell session, X11 forwarding, TCP/IP forwarding (channel multiplexing, pseudo terminals, flow control, remote program execution, authentication agent forwarding, terminal handling, etc.)

# The Components of the SSH suite

SSH binary programs, scripts

```
uhu:~> dpkg -L openssh-client | grep bin
/usr/bin
/usr/bin/ssh
/usr/bin/scp
/usr/bin/ssh-add
/usr/bin/ssh-agent
/usr/bin/ssh-keygen
/usr/bin/ssh-keyscan
/usr/bin/sftp
/usr/bin/ssh-vulnkey
/usr/bin/ssh-copy-id
/usr/bin/ssh-argv0
/usr/bin/slogin
uhu:~>
```

# The Components of the SSH suite

## SSH man page

```
uhu:~> ssh --help
usage: ssh [-1246AaCfGkKkMMNnqsTtVvXxY] [-b bind_address]
          [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option]
          [-L [bind_address:]port:host:hostport] [-i identity_file]
          [-c cipher_spec]
          [-D [bind_address:]port] [-e escape_char] [-F configfile]
          [-p port]
          [-R [bind_address:]port:host:hostport] [-S ctl_path]
          [-w tunnel:tunnel] [user@]hostname [command]

uhu:~>
```

# The SSH suite

## SSH parameters

### Parameter of SSH

- If command is specified, it is executed on the remote host instead of a login shell (s. example on next slide)
- default locations of configuration files
  - configuration file: `~/.ssh/config`
  - private key: `~/.ssh/id_rsa` `~/.ssh/id_dsa`
- Parameters:
  - `-l` username (ssh `-l sysadmin narwal`)
  - `username@hostname` (ssh `sysadmin@narwal`)
  - `-X` (X11 forwarding: ssh `-X sysadmin@popeye`)
  - `-N` do not execute command (is for port forwarding)
  - `-f` go into background
  - `-L` create tunnel (`-L 1025:homer.risc.uni-linz.ac.at:25`)
  - you can use more `-L` option in one command, (create more tunnels!)
  - `-D` works a SOCKSv5 proxy server (`-D 3000 narwal`)
  - `-v` Verbose mode to debug problems and see the progress of connection (`-vv`, `-vvv`)



# The SSH suite

## SSH examples

```
hades:sysadmin!8> ssh ke@bullfinch
ke@bullfinch's password:
Linux bullfinch 2.6.24-etchnhalf.1-686 #1 SMP Thu Nov 5 02:25:56 UTC 20
..... deleted .....
No mail.
Last login: Sat Nov 21 17:45:11 2009 from hades.risc.uni-linz.ac.at
Sat Nov 21 17:45:12 CET 2009
bullfinch>

hades:sysadmin!12> ssh gonzales who
cschneid pts/2      Nov 18 12:11 (ozelot.risc.uni-linz.ac.at)
cschneid pts/3      Nov 19 15:33 (ozelot.risc.uni-linz.ac.at)
cschneid pts/4      Nov 18 13:52 (ozelot.risc.uni-linz.ac.at)
cdoench  pts/5      Nov 20 09:50 (dog.risc.uni-linz.ac.at)
mkauers  pts/6      Nov 21 12:01 (fennek.risc.uni-linz.ac.at)
hades:sysadmin!13>
```

[Online presentation of the above commands](#)

# The SSH suite

ssh with command

## X11 forwarding wird activated

```
hades:sysadmin!13> ssh -X gonzales
Linux gonzales 2.6.26-2-amd64 #1 SMP Thu Nov 5 02:23:12 UTC 2009 x86_64
Last login: Fri Nov 20 15:24:10 2009 from tc14.risc.uni-linz.ac.at
gonzales:sysadmin!1>
gonzales:sysadmin!1> mathematica &
[1] 18455
gonzales:sysadmin!2> kill -TERM 18455
gonzales:sysadmin!3>
```

## Online presentation of invoking Mathematica remotely

# Firefox - Preferences - Advanced

## Network - Connection settings

### Manual proxy configuration

- To set: HTTP proxy host and port number
- Useful is the setting: SOCKS (v5) host and port number
  - use localhost and a port number and OpenSSH with the `-D portnumber` option

### SOCKS - Socket Secure

- is an Internet protocol that routes network packets between a client and server through a proxy server
- SOCKS server proxies TCP connections to arbitrary IP address
- v5 uses authentication
- for more info learn: [Wikipedia about openssh and SOCKSv5](#) and/or their man pages

# Firefox - Preferences - Advanced

## Network - Connection settings - SOCKS host and port

### SOCKS proxy settings

- Firefox acts as the client
  - SOCKS host (server): localhost (you start ssh locally)
  - port number: any number above 1024 (e.g. 4000)
- The OpenSSH program is a SOCKS server
  - creates an openssh connection from your laptop to any host where you have an account, e.g. to the RISC host roadrunner.risc.jku.at, with the already defined port number
  - (on your laptop, i.e. localhost) `ssh -D 4000 roadrunner.risc.jku.at`
  - ssh will log in roadrunner
  - roadrunner replaces all original IP numbers with it's own IP number
  - if you do not use VPN to RISC, this is a simpler (but not equivalent) solution
  - do not use it on a RISC computer ! All users on the same computer can connect to the given port ! Use it only on your laptop !

# Firefox - Preferences - Advanced - Network

## Proxy settings - Default

**Connection Settings**

**Configure Proxies to Access the Internet**

No proxy

Auto-detect proxy settings for this network

Use system proxy settings

Manual proxy configuration:

HTTP Proxy:  Port:

Use this proxy server for all protocols

SSL Proxy:  Port:

FTP Proxy:  Port:

SOCKS Host:  Port:

SOCKS v4  SOCKS v5

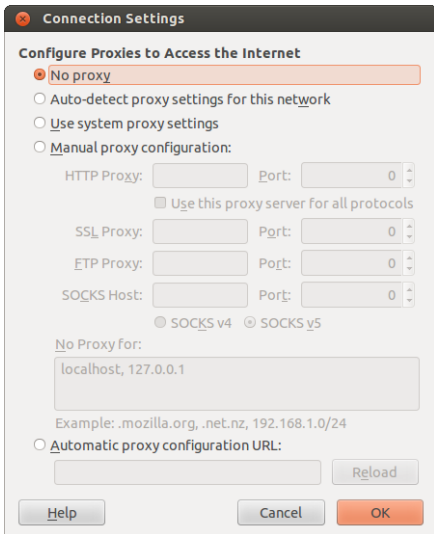
No Proxy for:

Example: .mozilla.org, .net.nz, 192.168.1.0/24

Automatic proxy configuration URL:

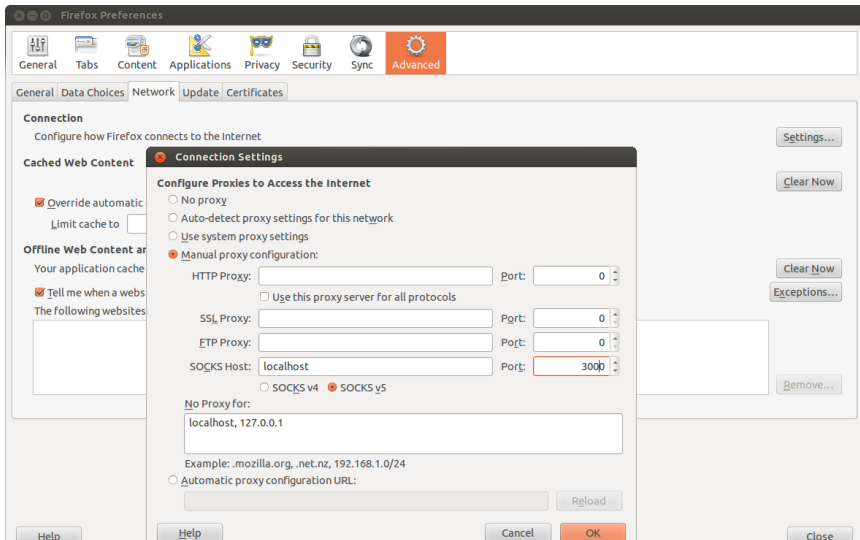
# Firefox - Preferences - Advanced - Network

Proxy settings - No proxy - direct connection to the Internet



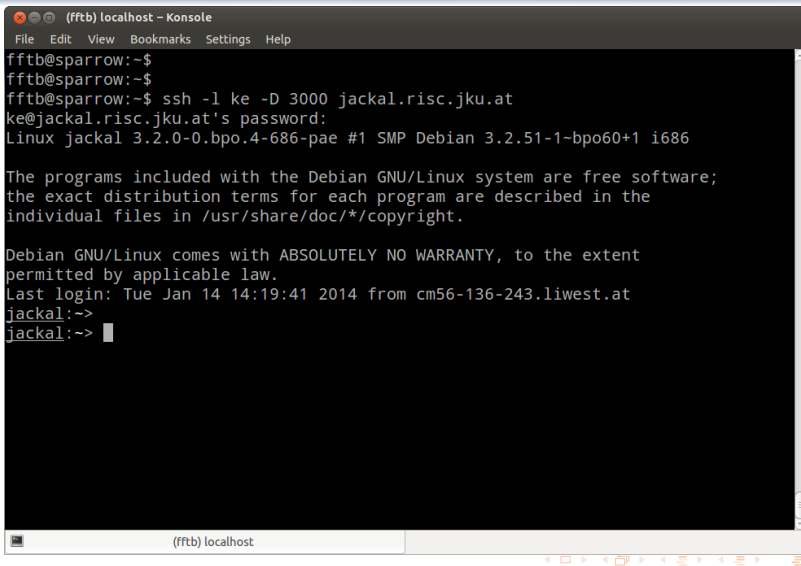
# Firefox - Preferences - Advanced - Network

## Proxy settings - SOCKS proxy settings



# Firefox - Preferences - Advanced - Network

Proxy settings - SOCKS proxy settings - the appropriate SSH command



```
(fftb) localhost - Konsole
File Edit View Bookmarks Settings Help
fftb@sparrow:~$
fftb@sparrow:~$
fftb@sparrow:~$ ssh -l ke -D 3000 jackal.risc.jku.at
ke@jackal.risc.jku.at's password:
Linux jackal 3.2.0-0.bpo.4-686-pae #1 SMP Debian 3.2.51-1-bpo60+1 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jan 14 14:19:41 2014 from cm56-136-243.liwest.at
jackal:~>
jackal:~> █
```



# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling**
- 5 SSH no password
- 6 VNC
- 7 X2Go
- 8 RDP

# SSH tunneling

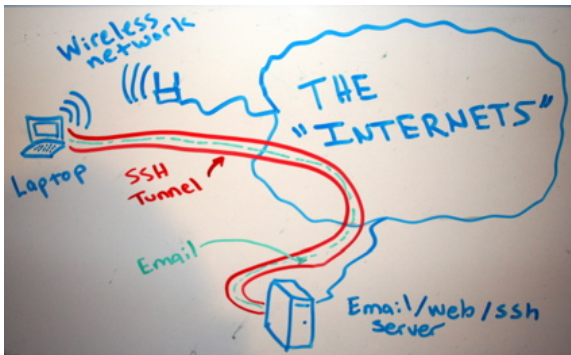
## What is an SSH tunnel

- tunnel is a networking term, means a connection, usually encrypted
- connects two computers together across a usually untrusted network

## Why do we need it - the Internet is very insecure !

- your laptop/home computer connects to another computer without encryption
- some protocols have encryption built in, some do not have
  - does have your email client? (pop3s, imaps, smtp)
  - your ftp program, VNC client, etc. does not have
- **Never use clear text connections !**
  - **definitively not for login/password data!**
- **always configure SSH tunnel for your connections!**

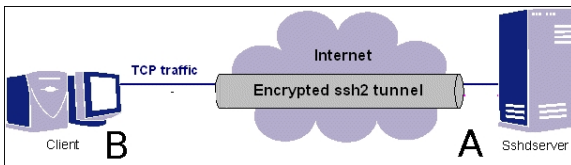
# Secure WLAN connection through the Internet



# SSH tunnel through the Internet

## SSH Tunnel Topology

- Client B (laptop, PC at home, etc) connects using local ports
- Server A running the sshd server program
  - mail server: port 25 smtp; VNC server: port 5901
- through an SSH tunnel - encrypted connection !



# How to make SSH tunnel in Linux

## basic version:

- **ssh -L localport:remote-hostname:remote-hostport remote-hostname**
  - Specifies that the given port (localport) on the local (the client) host is to be forwarded to the given remote host (remote-hostname) and remote port (remote-hostport). The remote host runs the sshd-server.
  - `ssh -L 22000:bullfinch.risc.uni-linz.ac.at:143 bullfinch.risc.uni-linz.ac.at`
- **ssh -L localport:hostname:hostport remotehost**
  - Specifies that the given port (localport) on the local (the client) host is to be forwarded to the given host (hostname) and port (hostport) on the remote side (remotehost).
  - `ssh -L 20000:grizzly.risc.uni-linz.ac.at:143 bullfinch.risc.uni-linz.ac.at`
  - **hostname** and **remotehost** may be different !
  - the connection from **localhost** to **remotehost** is secure
  - the connection from **remotehost** to **hostname** is unsecure !!

# How to make SSH tunnel in Linux

## full version

- `ssh -f -N -L localport:host:hostport sshd-server-computer`
  - B: local computer, C: host, A: sshd-server-computer
- -N is for portforwarding (do not execute command)
- -f go into background
- you can use more -L option in one command, (create more tunnels!)

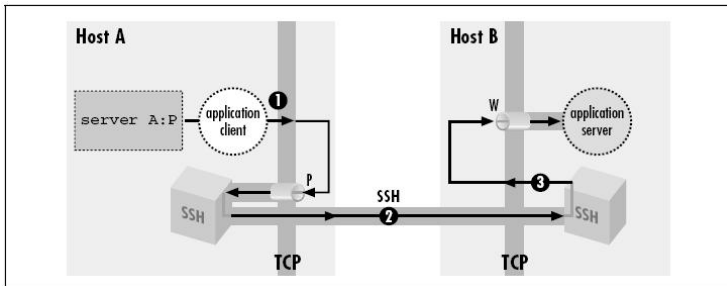


# SSH Tunnel - Port forwarding

Windows [www.putty.org](http://www.putty.org)

## Port forwarding

- `host(1): ssh -f -N -L P:B:W sshd-server-computer(3)`



# SSH Tunnel - Port forwarding

examples for more tunnels

## Tunnels for giraffe (VNC) and crutch (RDP)

- shell aliases: giraffe and crutch

```
sparrow:~> which giraffe
```

```
giraffe:  aliased to
```

```
ssh -f -N -L 5901:localhost:5901 giraffe.risc.uni-linz.ac.at
```

```
sparrow:~>
```

```
sparrow:~> which crutch
```

```
crutch:  aliased to
```

```
ssh -f -N -L 3389:crutch.risc.uni-linz.ac.at:3389
```

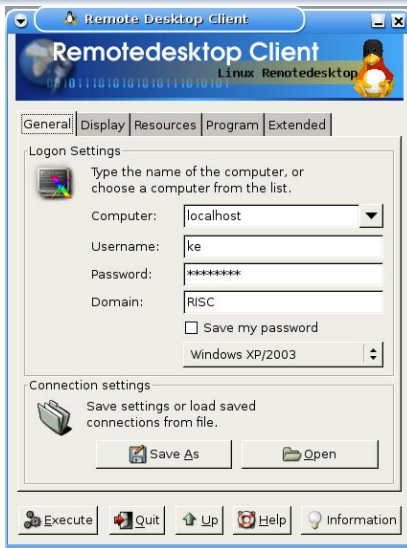
```
gorilla.risc.uni-linz.ac.at
```

```
sparrow:~>
```



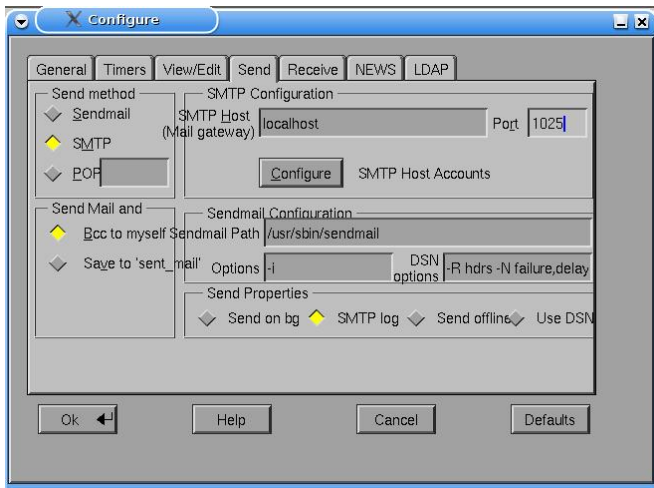
# SSH Tunnel - Port forwarding

examples: rdp tunneling from laptop to crutch though homer



# SSH Tunnel - Port forwarding

examples: sending email with stmp by tunneling from laptop to homer



# SSH Tunnel - Port forwarding

examples for more tunnels

## Tunnels

- shell alias: bt (bullfinch tunnel)

```
uhu:~> which bt
```

```
bt:      aliased to
```

```
ssh -f -N
```

```
-L 20000:grizzly.risc.uni-linz.ac.at:143
```

```
-L 22000:bullfinch.risc.uni-linz.ac.at:995
```

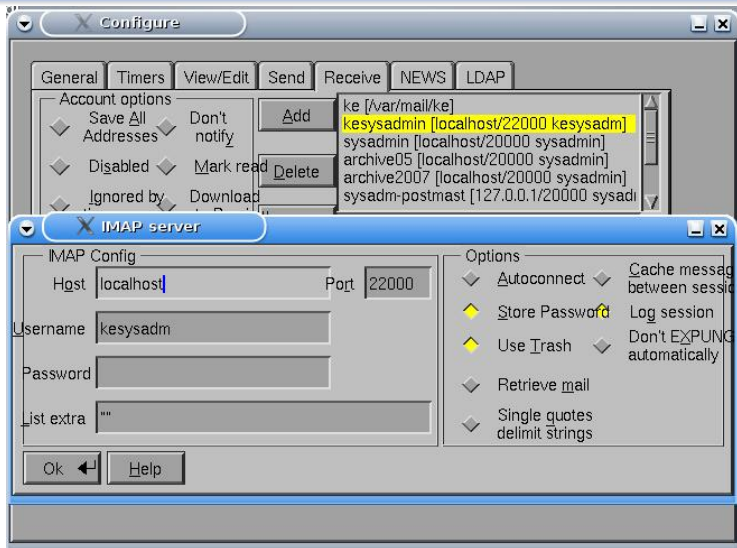
```
-L 1025:homer.risc.uni-linz.ac.at:25
```

```
bullfinch.risc.uni-linz.ac.at
```

- sysadmin:imap, kerdei:pop3s kerdei:smtp

# SSH Tunnel - Port forwarding

examples: more IMAP connections through the same tunnel



## SSH Clients - MS Windows

### Use open source SSH programs

- [www.putty.org](http://www.putty.org)
  - you can configure ssh tunnels with putty
- [www.winscp.net](http://www.winscp.net)

# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling
- 5 SSH no password**
- 6 VNC
- 7 X2Go
- 8 RDP

# Remote login without passwd by SSH

How to set up

## Basics of the authentication

- SSH authentication methods
  - password authentication; private key authentication
- private key authentication
  - Create a private key - public key pair with **ssh**; set the passphrase for the private key !
  - Copy the public key to the remote computer
  - Configure the authentication agent: **ssh-agent**
  - use **ssh-add** command to add your identity to the ssh-agent
- Customizing the authentication
  - installing **ssh-askpass**
  - Starting ssh-add by an icon

# Remote login with SSH

create a public key

## Create public key

- Create a public key: `ssh-keygen -t rsa`
  - **always USE a passphrase**
  - without passphrase: if your private key is stolen your identity is stolen
  - choose it different from your password, choose a long one
  - it must as save as your password, it can be more save (less restriction)

```
sparrow:~> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ke/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ke/.ssh/id_rsa.
Your public key has been saved in /home/ke/.ssh/id_rsa.pub.
The key fingerprint is:
5f:cb:bf:77:17:eb:82:f7:35:a1:c3:eb:32:d7:bd:af ke@sparrow
The key's randomart image is:
+--[ RSA 2048 ]-----+
```





# Remote login with SSH

copy public key

## Copy public key

- copy the public key to the RISC computer and add to `.ssh/authorized_keys` file

```
sparrow:~> cat .ssh/id_rsa.pub |  
ssh narwal.risc.jku.at 'cat - >> .ssh/authorized_keys'
```

```
ke@narwal.risc.jku.at's password:  
sparrow:~>
```

- you will be asked for your password on the remote computer
- check that it works:
  - `ssh -X narwal.risc.uni-linz.ac.at`
  - passphrase will be asked for

## Live demonstration now

# Remote login with SSH

ssh-add ssh-agent

## ssh-agent

- Authentication agent, **ssh-agent**
  - saves the identity value (private key) in the memory
  - supports authentication requests from SSH
  - started by login in KDE, GNOME

## ssh-add

- transfers the identification (.ssh/id\_dsa) to ssh-agent
- asks for the passphrase, to decrypt the private key

```
sparrow:~> ssh-add .ssh/id_rsa
Enter passphrase for .ssh/id_rsa:
Identity added: .ssh/id_rsa (.ssh/id_rsa)
sparrow:~>
```

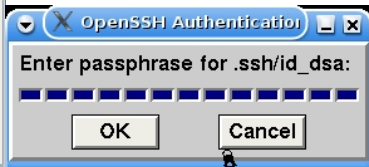
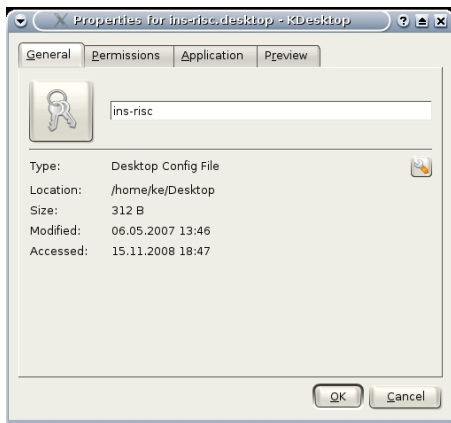
- will invoke ssh-askpass, if get a zero in standard input

# Customizing ssh-add icon for ssh-add

Create a small script in i.e. `/usr/local/bin/` or `~ /bin`

```
#!/bin/csh
```

```
cat /dev/null | ssh-add .ssh/id_dsa
```



# Important security tips

for using the Internet

## for more security in the Internet

- always use Linux and not MS Windows
  - Linux is much more secure, open source, etc.
  - Debian is **more** secure as Ubuntu (testing period half year only)
- never use Windows for internet banking
- never use smartphones for getting the mobile TAN
  - use old simple mobile phones to get mobile TAN
- always transport your identification on a secure way
  - through secure intranet (RISC) or by USB Sticks

## The most UNSECURE internet mediums are:

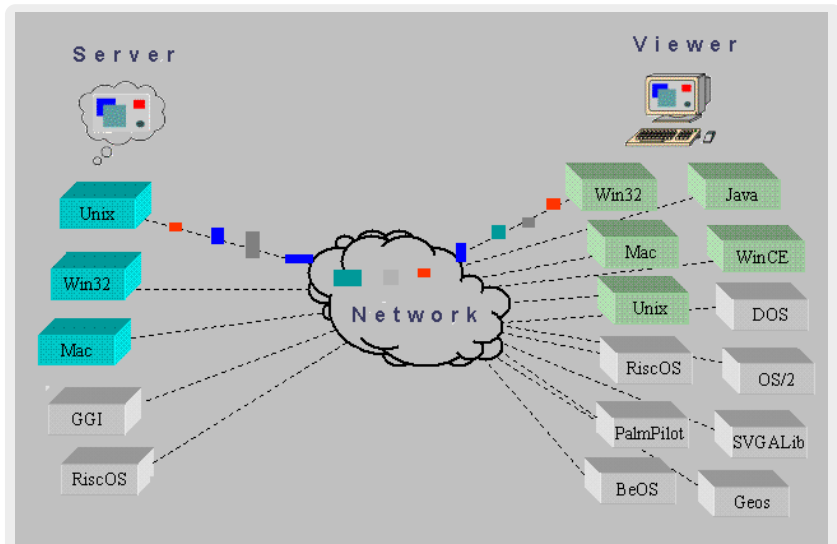
- public WLAN hotspots
- cable networks of the internet providers

# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling
- 5 SSH no password
- 6 VNC**
- 7 X2Go
- 8 RDP

# VNC (Virtual Network Computing)

referred as Virtual Network Console, too



# VNC - Virtual Network Computing

## Basic Features

### VNC is a free platform-independent application

- is a Client-Server architecture based on the RFB protocol
- is a graphical desktop sharing system
  - without the need of X on the client side
- transmits the keyboard and mouse events from one computer to another
- relays the graphical screen updates back in the other direction
- is **not a secure** protocol
  - although passwords are not sent in plain-text
  - crack could be successful if both the encryption key and encoded password are sniffed from a network
- always use VNC through an **SSH tunnel** !
- Open source tool: <http://www.realvnc.com>

# VNC - Virtual Network Computing

## Basic terminology

### Framebuffer (FB)

- is a video output device that drives a video display from a memory buffer containing a complete frame of data
- the information in the buffer consists of color values for every pixel on the screen
- total memory required for the FB depends on the resolution, and on the color depth
- a FB device driver was created for X11: XF86 FBDev as standard part of XFree86
- FBDev is basic driver in X, without using the features of the GPU



# VNC - Virtual Network Computing

## RFB Protocol

### Remote Framebuffer (RFB) protocol

- is a simple protocol for remote access to graphical user interfaces
- it works at the framebuffer level, it is applicable to all windowing systems and applications, including X11, Windows and Macintosh.
- seamless cross-compatibility
  - between the many different VNC client and server implementations
- clients and servers negotiate using
  - the best RFB version
  - most appropriate compression and security options

### RealVNC, Ltd.

- continues development of VNC and to maintain the RFB protocol

# VNC - Virtual Network Computing

## VNC Server

### VNC Server features

- runs on the **remote** computer !
- does not have a physical display! (does not bind to a display)
- consists of **two** servers on Linux/Unix OS
  - Framebuffer Server: to communicate **remotely** with the VNC client
  - X Server: to communicate **locally** (on the remote computer) with the X-clients, presenting itself as a real X-Server
  - **the X-server part fills up the framebuffer with the output from the X-clients**
  - **the FB-server part transfers the content of**
  - **the FB to VNC-client(s)**
- the session information will be kept in the server side
  - if you disconnect from the VNC server it will **not** close the session
  - Disconnecting from VNC server behaves like locking the session and switching off the monitor
- you have explicitly kill the VNC server after your work !

# VNC - Virtual Network Computing

## VNC Server II

### VNC Server features

- by default uses TCP ports 5900 through 5906
  - each port corresponds to a separate screen (:0 to :6)
- uses ports 5800 through 5806 for java connections
  - allowing clients to interact through a Java-enabled web browser
  - (be careful using Java - security holes)
- Xvnc is the Unix VNC server, it is based on standard X server
- any number of Xvnc server can be started (think on resources!)
  - choose a simple desktop - save resources
- more clients can connect to the same server
- VNC need more/high bandwidth because of transferring screenshots

# VNC - Virtual Network Computing

## Starting the VNC Server

### Starting the VNC server

- log in by **ssh** to a RISC computer, e.g. beagle:
  - `ssh -l username beagle.risc.uni-linz.ac.at`
  - `uhu> ssh -l username beagle.risc.uni-linz.ac.at`
- start the VNC server by the command:
  - `beagle:1> vncserver -geometry 1024x768 -depth 24`

- You will see something similar in the screen (it just ask a session password at the first run):

```
You will require a password to access your desktops.
```

```
Password:
```

```
Verify:
```

```
New 'X' desktop is beagle:1
```

```
Starting applications specified in /etc/X11/Xsession
```

```
Log file is /home/yourusername/.vnc/beagle:1.log
```

- The VNC server password must be same secure as your login password ! It gives access to your home directory.

# VNC - Virtual Network Computing

## Starting the VNC server

### Starting Server

- You have to memorize the server name and the screen number - after the computer name (in this case it is ":1")
  - The port number will be 5901 (5900+screen number)
- You have to **shutdown** the VNC server, after you do not need it:
  - beagle:3> **vncserver -kill :1**  
Killing Xvnc4 process ID 2693  
beagle:4>
- The configuration and log data for the VNC server are stored in the:
  - /home/<username>/vnc/ directory
- The VNC server asks for the password at the first time only
- If you forgot the password for the VNC server, remove or change it:
  - **rm /home/<username>/vnc/passwd**
  - **vncpasswd /home/<username>/vnc/passwd**

# VNC - Virtual Network Computing

## Starting the VNC client

### Starting the VNC Client

- create an ssh tunnel on your local computer to the vnc server:

```
ssh -f -l username -N -L 5901:localhost:5901 srvname
```

```
uhu> ssh -f -l username -f -N -L 5901:localhost:5901 beagle.risc.uni-linz.ac.at
```

- start the VNC client on your local computer

```
uhu> xvncviewer localhost:1
```

- best solution is to use a shell alias, e.g. for the tcsh in `/home/username/.cshrc` :

- beagle alias "ssh -f -l username -N -L 5901:localhost:5901 beagle.risc.uni-linz.ac.at "

- **source** /home/username/.cshrc

- activate the tunnel in the command line by **beagletunnel**

- Security Risk

- your password can be stolen using xvncclient without ssh tunnel !

- hacker get full access to your home directory

# VNC - Virtual Network Computing

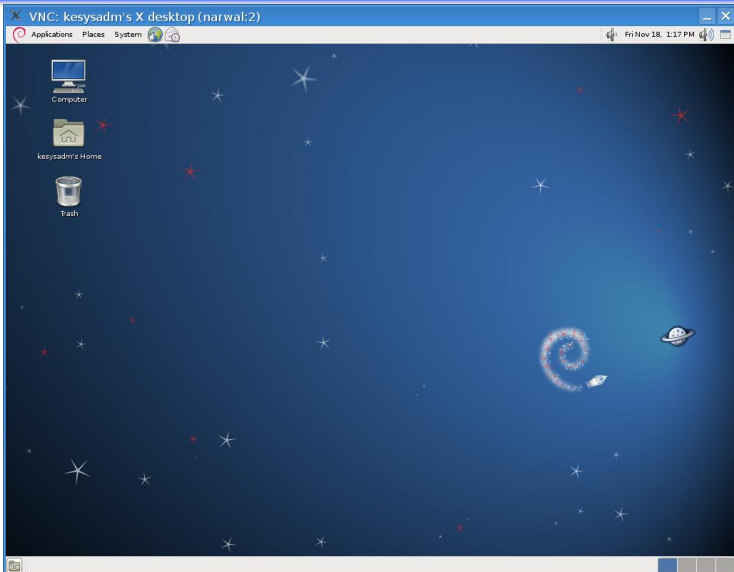
## VNC server and client starting

### Configuration of the vncserver at RISC

- the vncservers at RISC are configured with option **-localhost**
  - this means, that the vncserver accepts connections only from localhost (127.0.0.1)
  - with other words: you **MUST** use ssh tunnel to the host where the vncserver is running (otherwise you'll get error: **connection refused**).
- example: assumed, you started the vncserver on the computer gonzales.risc.uni-linz.ac.at, you need the following ssh-tunnel:
  - **ssh -f -l username -N -L 5901:localhost:5901 gonzales.risc.uni-linz.ac.at**
  - localhost will be replaced by 127.0.0.1, and this is the IP from which the vncserver accepts connections.

# VNC - Virtual Network Computing

## VNC Client - xnvviewer





# VNC - Virtual Network Computing

VNC Server - default xstartup file

xstartup file: `/home/user/.vnc/xstartup`

```
#!/bin/sh
```

```
xrdb $HOME/.Xresources
```

```
xsetroot -solid grey
```

```
#x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desk
```

```
#x-window-manager &
```

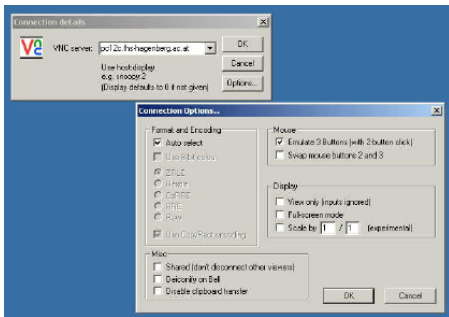
```
# Fix to make GNOME work
```

```
export XKL_XMODMAP_DISABLE=1
```

```
/etc/X11/Xsession
```

# Real VNC

## Using VNC under MS Windows



# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling
- 5 SSH no password
- 6 VNC
- 7 X2Go
- 8 RDP

# X2Go client - server software

## X2go

- X2Go is an implementation of NX remote desktop protocol
  - contains both client and server software package
  - remotely accesses the graphical desktop of a server
  - server package must be installed on a Linux server
  - client package runs on Linux, Windows, or MacOS
- Features
  - It is open source software
  - session resuming
  - low bandwidth support
  - session brokerage support
  - client-side mass storage mounting support
  - client-side printing support
  - audio support
  - authentication by smartcard and USB stick, too !

# NX protocol

## NX - Remote Desktop Protocol

- handles remote X Window System connections
- attempts to greatly improve the performance of the native X protocol
- it wraps remote connections in SSH sessions for encryption
- designed primarily to optimize X11 sessions
- NX server can be configured as a proxy server to tunnel:
  - Windows RDP sessions
  - VNC sessions

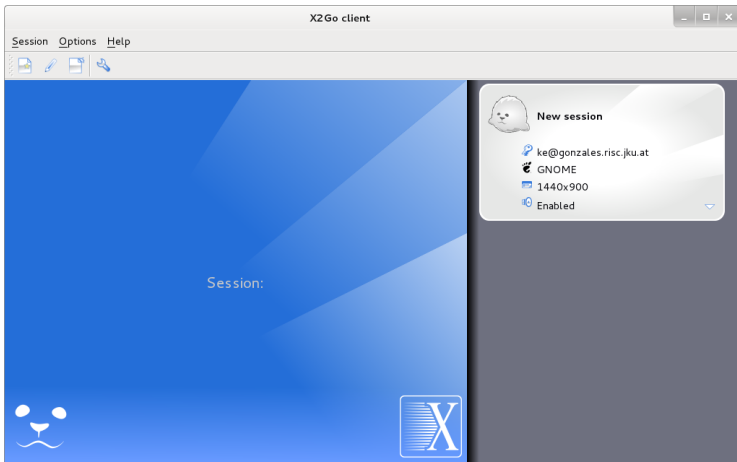
# NX security

## NX security

- the initial login between client and server happens using a DSA key-pair
- the public key is provided by the installation of the server
- the private key is distributed together with NX Client
- NX establishes an SSH secure channel once the server has authenticated the client
  - authentication of the user on the remote system happens on this channel
  - tunnels all the session traffic over the SSH channel

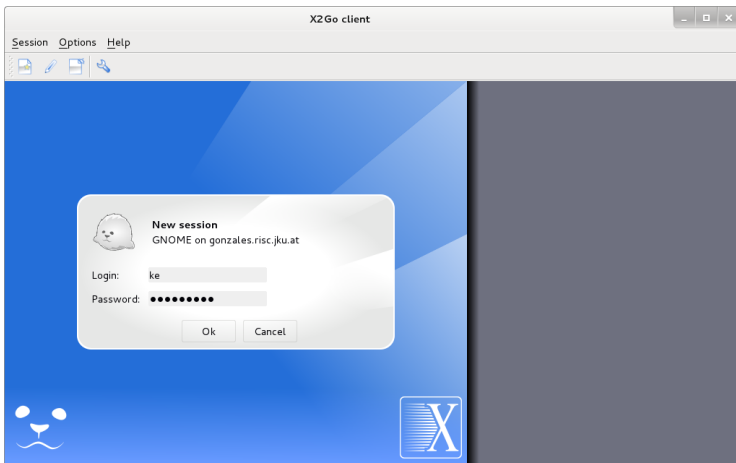
# X2go

## Starting



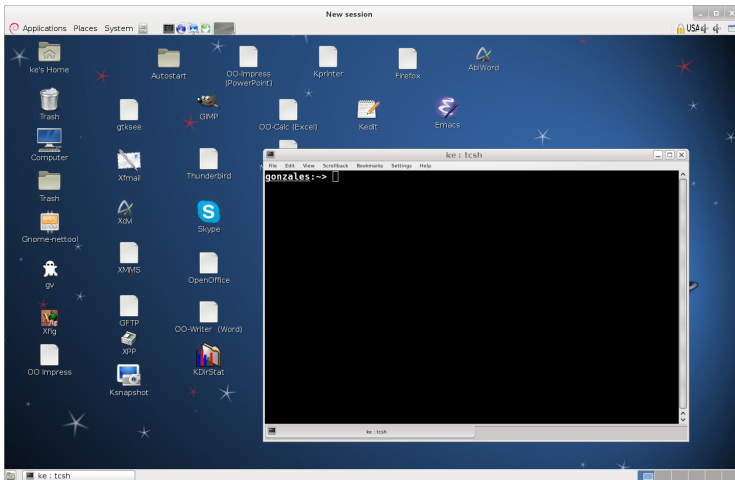
# X2go

Log in



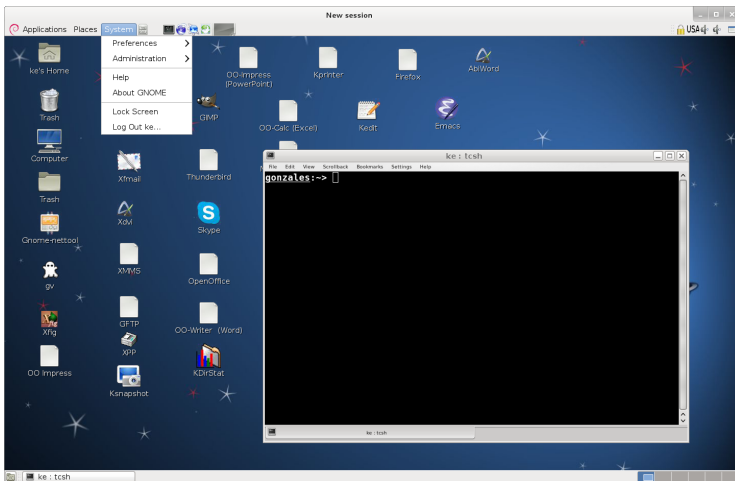


# X2go Running



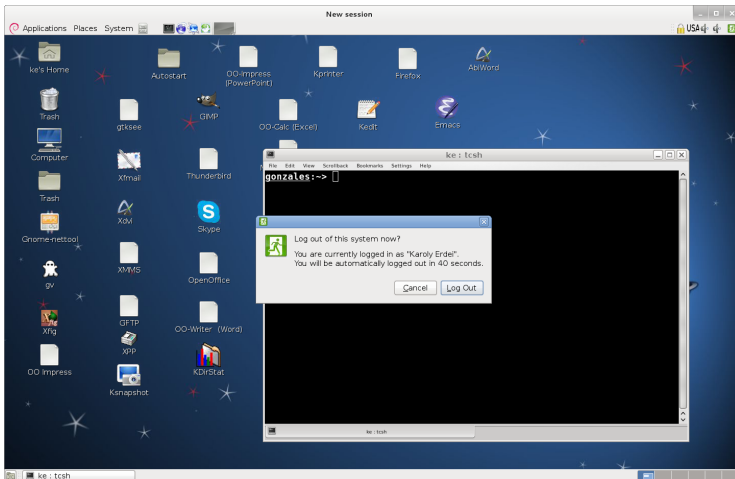
# X2go

Log out



# X2go

## Finish



# Agenda

- 1 Remote Login
- 2 File Services
- 3 Secure Shell
- 4 SSH tunneling
- 5 SSH no password
- 6 VNC
- 7 X2Go
- 8 RDP

# Remote Desktop Protocol

Microsoft Windows

## Windows NT/2000: Terminal Services extension

- Remote Desktop Protocol (RDP) developed in the mid 1990's by Microsoft
  - RDP client computer (Windows/Unix) opens a remote desktop session on a Windows NT/2000 server with terminal services extension
  - In client window, user sees another desktop running on the server
  - Introduced by Windows NT Terminal Server Edition
  - Installed at RISC in 1999 for MS Office Compatibility goals
  - The first MS Windows Multiuser OS !
- Windows XP:
  - Provides builtin RDP service functionality
- Windows 2003 Server: successor of NT/2000 Terminal Server Editon

# Remote Desktop Protocol

crutch - the RISC Windows 2003 server

## crutch: Linux - Windows integration

- Supporting the RISC users for some MS Windows applications
  - for software available only on MS Windows
- Microsoft Software
  - OpenOffice and MS-Office are not fully compatible
  - MS Office is available in the (near) last version on crutch
- Adobe Software
  - Adobe Acrobat 9 Pro Extended (2 concurrent licenses)
  - Adobe Photoshop Lightroom 2.1 (1 concurrent license)
- Other Software
  - ACDSee 8 (image management and manipulation sw)
  - Canon DPP (Digital Photo Professional, for Canon DSLR RAW images)
- Configuration of crutch
  - the riscwide home directory is available (scratch,too)

# Remote Desktop Protocol

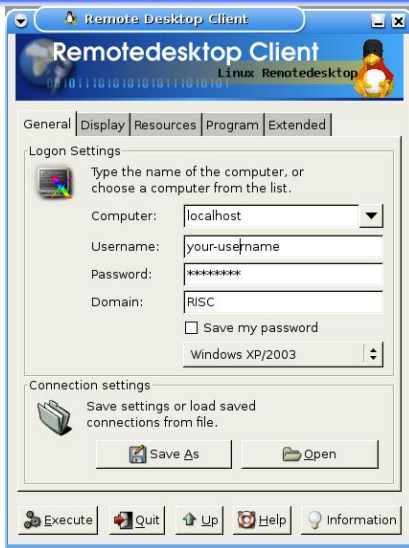
crutch - the RISC Windows 2003 server

## RDP ports, connections

- How to connect through an SSH tunnel to crutch
  - RDP uses the port 3389
  - the Windows-2003 server has no SSH server implementation
  - you have to connect to a Linux computer at RISC with SSH and make the tunnel through this computer to crutch
- `ssh -l username -f -N -L 3389:crutch.risc.uni-linz.ac.at:3389 beagle.risc.uni-linz.ac.at`
  - this is an SSH connection from your computer to beagle
  - the tunnel runs from your computer through beagle to crutch
  - the tunnel section between beagle and crutch is not secure
- Configuration of **grdesktop**
  - define **localhost** in the General options for the field Computer

# GRDesktop - Configuration

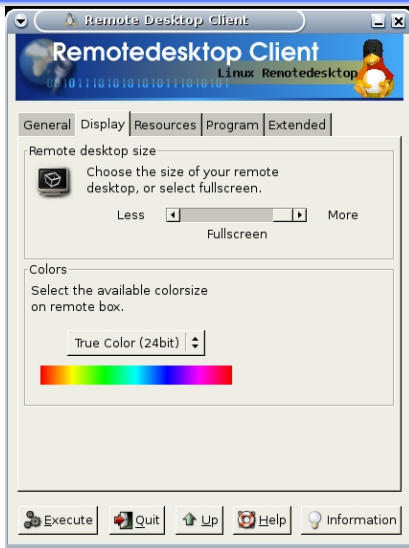
## Gnu RDP Client





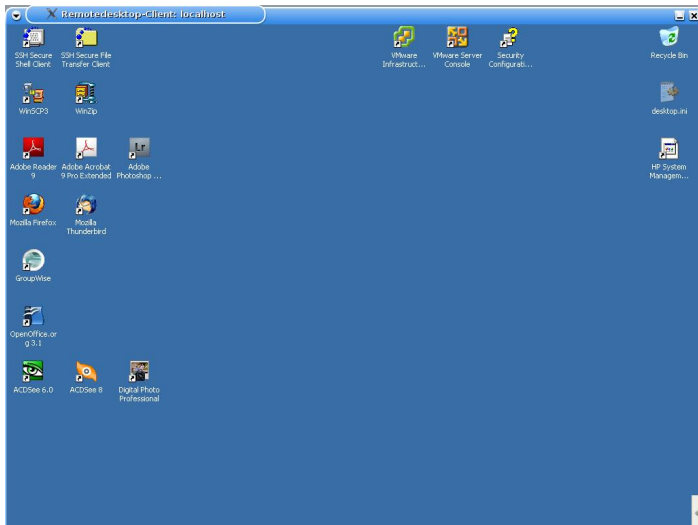
# GRDesktop - Configuration

## Gnu RDP Client



# GRDesktop

## Main screen



## End of Remote Services, Desktops

Thanks for your attention !