Програмиране в UNIX среда

Основи на системната администрация

Linux System Administration

SSH

SSH


SSH!
DON'T GIVE AWAY YOUR PASSWORD TO
ANYONE WHO MIGHT BE LISTENING!
Always use SSH (Secure SHell) when connecting
to Computer Science Department hosts, instead of
telnet or rsh. For more information:
http://www.cs.umd.edu/faq/ssh.html

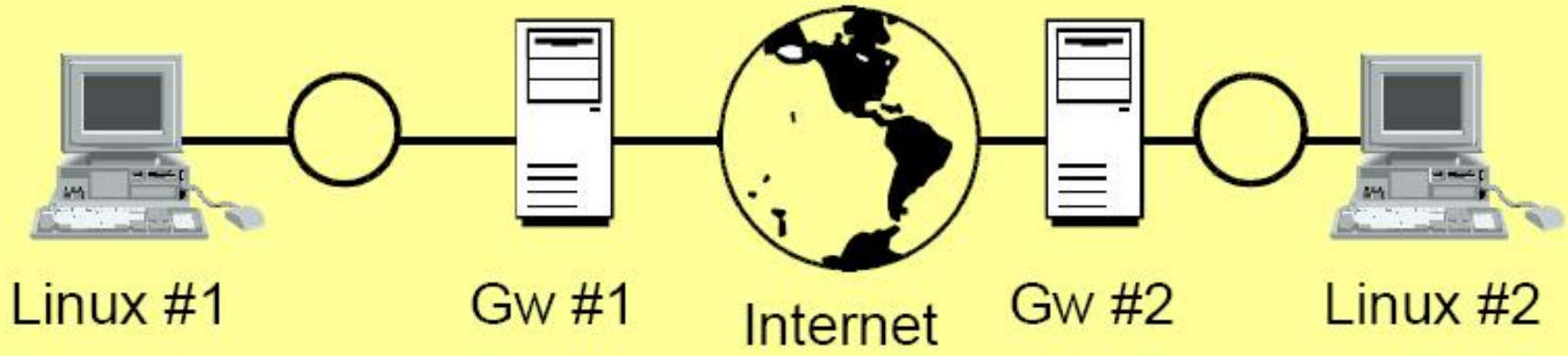
SSH

- Защо SSH? Проблеми с Telnet & Friends
- Кратко описание на SSH протоколите
- Получаване и инсталация на OpenSSH
- X11 Forwarding и Port Forwarding
- SSH агент
- scp, rsync през SSH
- Firewall Busting
- SSH vs. IPSec и други

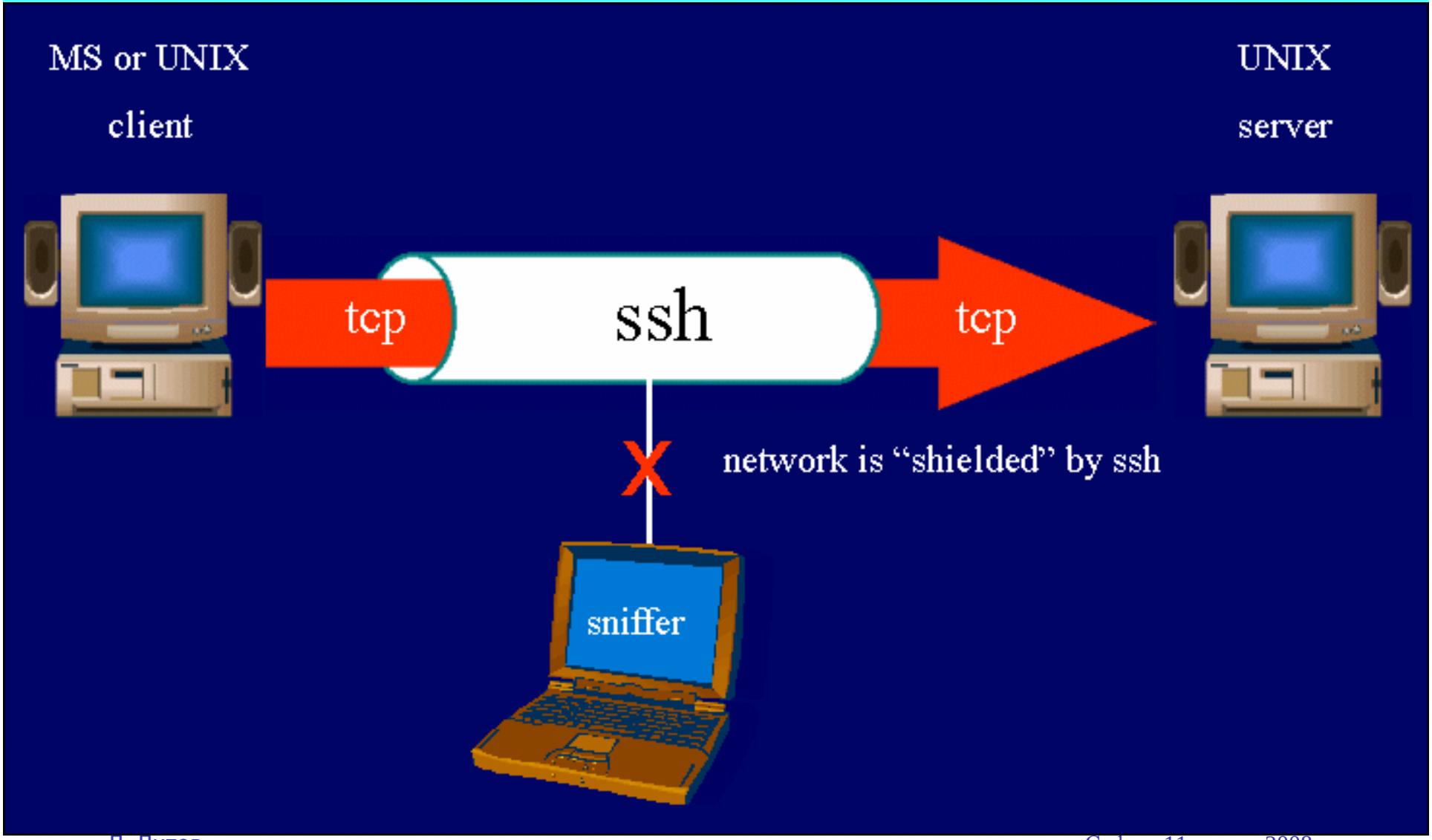
Защо SSH ?

- Do you care at all about privacy and security?
- Then *don't* use Telnet, rsh, rlogin and friends at all!
- Telnet: Clear−text passwords, clear−text session.
- rsh/rlogin: Even worse hostname−based trust mechanism is trivial to spoof. (Think /etc/hosts.equiv and ~/.rhosts)

Пример

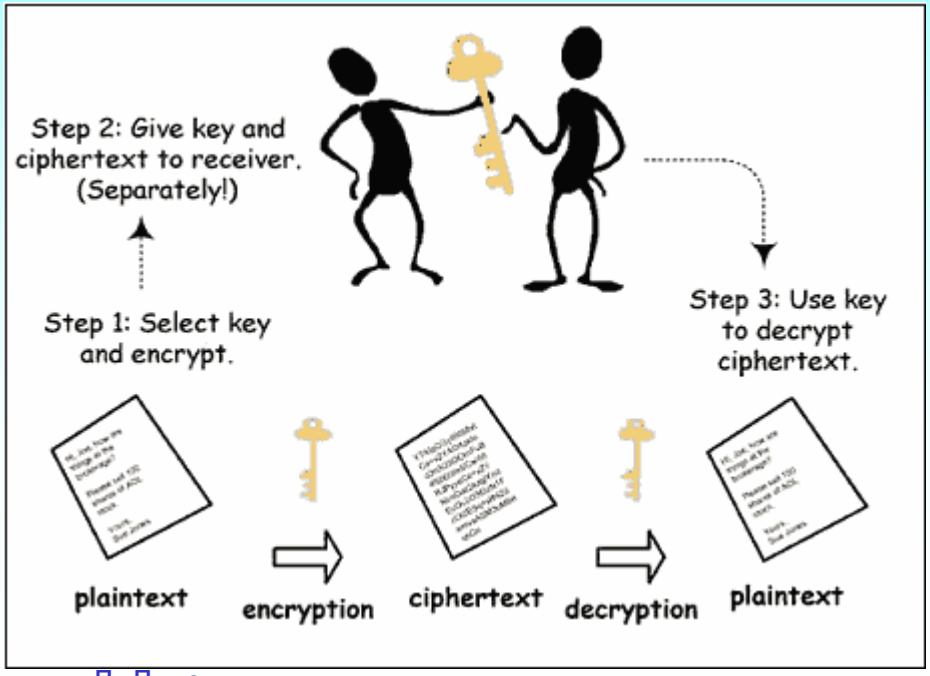


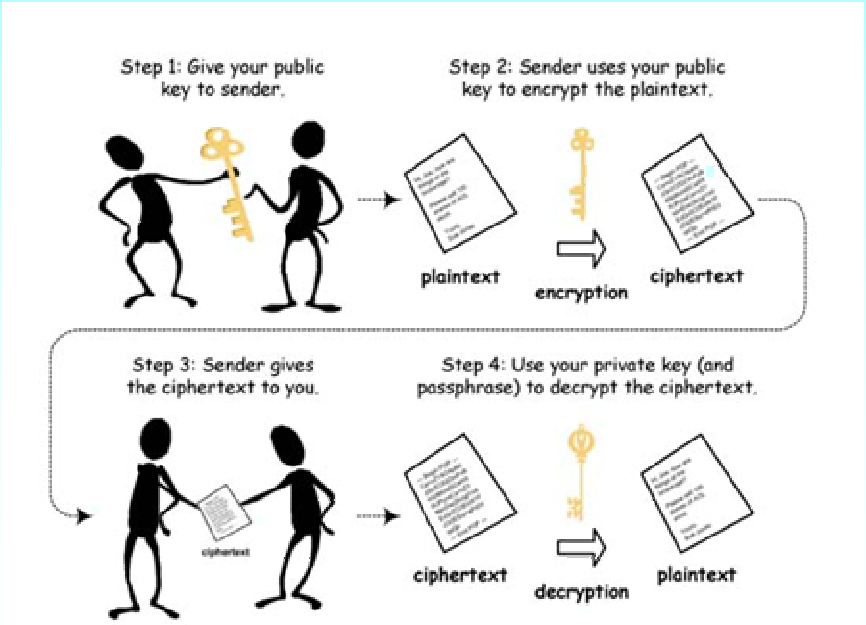
- If Linux #1 needs a connection to Linux #2, attackers can sniff packets on the Internet, on LAN #1, on LAN #2 or on either gateway.
- Therefore, we need a protocol which assumes eavesdroppers hear everything, but still cannot impersonate either side.
- The Secure Shell (SSH) protocols offer this capability.


MS or UNIX
client
UNIX
server
tcp
ssh
tcp
X
network is “shielded” by ssh
sniffer

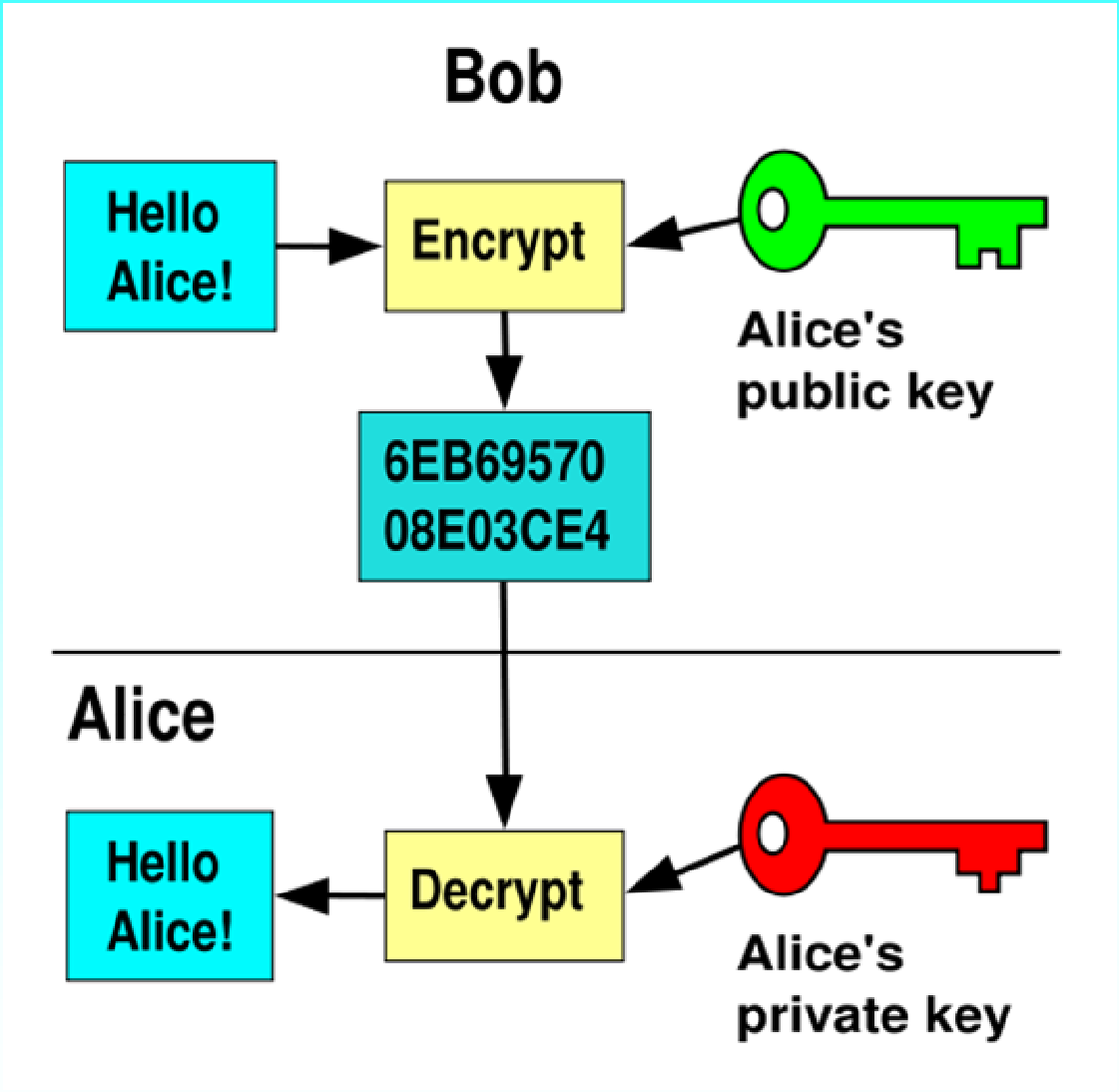
Криптография

- Symmetric Encryption: The *same* (secret) key is used for encryption and decryption. Ideally, arbitrary amounts of *chosen plaintext* and corresponding ciphertext will not reveal key. Symmetric encryption fast.
- Public Key Encryption: A *public* key is used for encryption and a secret *private* key for decryption. Or, the secret key for signing and public key for validation. Public key encryption slow.





Криптиране и декриптиране с публичен ключ



SSH1 протоколът (накратко)

- The server has a public/private key pair.
- The client *must know* the server's public key in advance.
- The server sends its public key to the client as well as a periodically–generated server key. Client verifies that public key is known.
- The client generates a random *session key*, encrypts it with the host and server key, and sends it to the server. Everything is now encrypted with the session key.

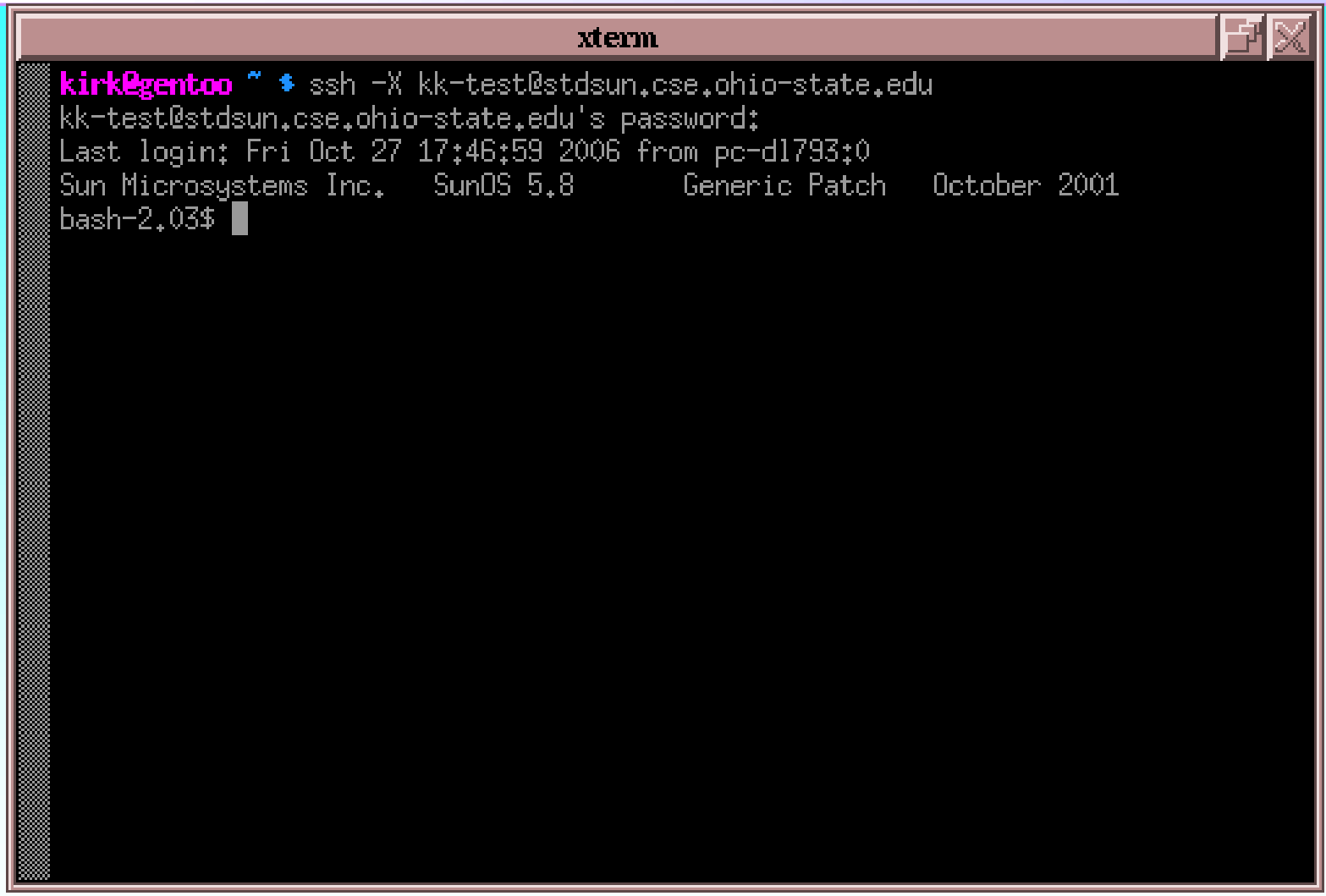
SSH2 протоколът (накратко)

- One of a number of *key−exchange* algorithms is run. At the end, client and server share a secret key, unknowable by eavesdroppers.
- Digital signatures verify identity of server to client.
- Everything following key exchange is encrypted with the shared secret.

Получаване и инсталиране на SSH

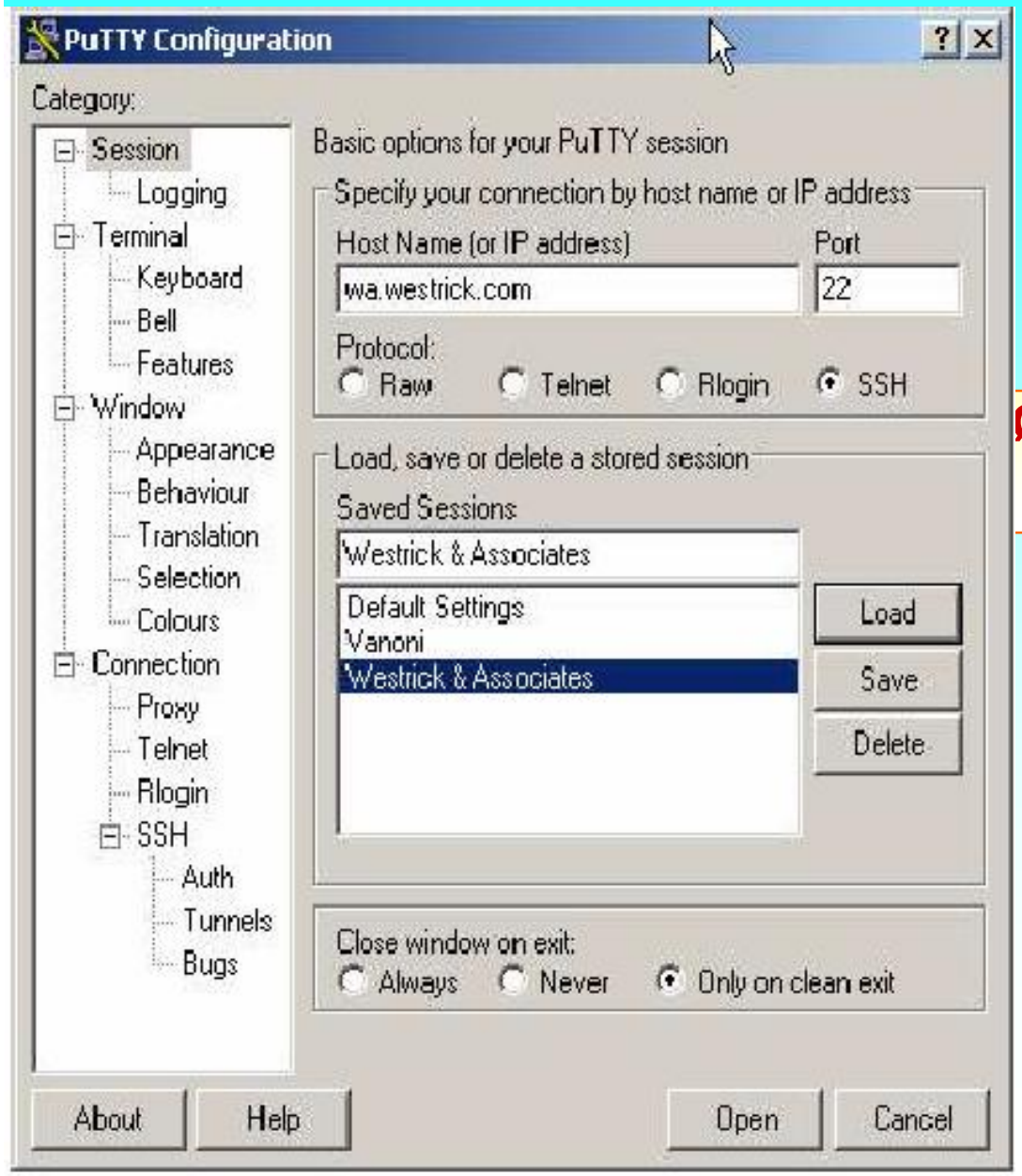
- Best to use OpenSSH. It’s free and developed by OpenBSD developers who are security fanatics.
- Go to http://www.openssh.com and follow the links to "portable OpenSSH". There are Linux RPM’s available.
- You also need OpenSSL, available from the OpenSSH download sites.

Screenshot of SSH in Action



- As simple to use as rsh!
- Just use *ssh host*, enter passphrase and you have a shell.

www.chiark.greenend.org.uk/~sgtatham/putty/



Ø PuTTY: A Free Telnet/SSH Client

Verify the Host Key

- If SSH does not recognize the host key, it will show the *key fingerprint* and ask if you want to continue.
- ***DO NOT*** continue unless you are absolutely sure the key fingerprint is correct.
- If SSH gets a different key than the one in its known_hosts list, it will print a huge warning and refuse to continue. Getting the wrong host key is usually because someone messed up, but could be due to spoofing.

Setting up the SSH Client

- Generate an SSH key pair: *ssh−keygen*
- Enter a pass phrase to protect the private key.
- Copy the private key to ~/.ssh/identity, mode 0600.
- Copy the public key to the remote machine in ~/.ssh/authorized_keys.
- You can also use "encrypted password authentication", but this is *not recommended.*

Password Authentication

- Just like Telnet or login, except username and password are encrypted.
- Advantage: Don’t have to generate a key pair.
- Disadvantage: Less secure. Susceptible to password−guessing attacks.

Public Key Authentication

- Uses public/private key pair for authentication.
- Disadvantage: Have to generate a key pair and put the public key in ~/.ssh/authorized_keys.
- Advantage: Defeats password−guessing attacks unless attacker has access to private key.
- Key pairs can optionally be restricted in capability. For example, one key could be limited to running a "tar" command for backup.
- Allows fine−grained access control.

X11 Forwarding

- SSH gives you an *encrypted pipe* through the Internet.
- Usually, this pipe is used for interactive shell sessions.
- However, SSH can also do *X11 Forwarding*.
- On the server side, the SSH server creates a "fake" X server (for example, remotehost:10).
- X connections to that server are forwarded through the encrypted pipe.

X11 Forwarding

- When the SSH client sees a forwarded X connection coming through, it opens a connection to the real X server and forwards X traffic.
- Net result: You can remotely run X applications, and all X traffic is securely encrypted.
- X forwarding can be disabled by the client or the server.

Port Forwarding

- SSH can forward arbitrary TCP ports over the encrypted pipe.
- Two flavours: Forwarding of local (client−side) ports and forwarding of remote (server−side) ports.
- Example: **ssh −L 8080:remotemach:80**
- On the client, TCP port 8080 is forwarded through the encrypted pipe to port 80 on remotemach.

Port Forwarding

- **ssh −L 8080:remotemach:80**
- SSH client listens on port 8080 on 127.0.0.1
- When an incoming connection arrives, client notifies the server of this fact. Server opens a connection to remotemach, port 80.
- All further traffic is forwarded over this encrypted pipe.
- If the ssh server is a gateway, remotemach need not even have a routable IP address. It just has to be reachable from the ssh server.

Forwarding Remote Ports

- **ssh −R 8080:localmach:80**
- SSH server listens on port 8080 on 127.0.0.1.
- When an incoming connection on port 8080 arrives, server notifies the client of this fact. Client opens a connection to localmach, port 80.
- All further traffic is forwarded over this encrypted pipe.

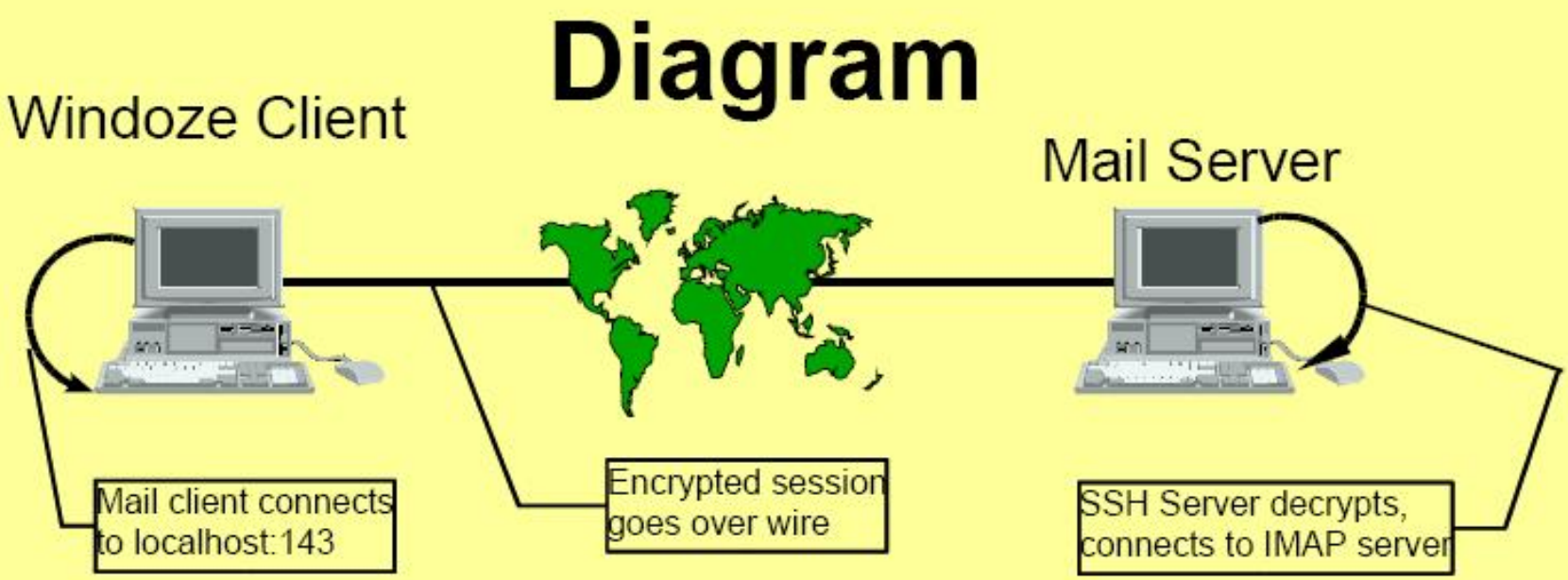
Port Forwarding Caveats

- Only *root* can port−forward privileged local ports.
- Forwarded ports only listen to 127.0.0.1 by default. This is a security feature (which can be overridden.)
- Only *root* on the remote end can forward *from* privileged remote ports. Anyone can forward *to* privileged ports.

Nice Use of Port Forwarding

- Secure access to IMAP or POP3 servers, especially for Windoze clients.
- Using a free Windoze SSH client, set up port−forwarding from local ports 25 and 143 to corresponding ports on mail server.
- On mail server, the only port open (for remote access) is SSH.
- Port−forwarding takes care of restricting access to IMAP, encryption and MTA relaying configuration.

Поща



- Set up Windoze mail client to use 127.0.0.1 as incoming/outgoing mail server. :−)
- Wait−a−minute! Only *root* can forward privileged ports...
- On Windoze, everyone is *root*...

SSH Agent

- If you use a passphrase for your private key (recommended!), it's annoying to have to type it in each time.
- *Ssh−agent* lets you enter your passphrase once per session (e.g., at the start of an X session) and then decrypts and remembers your key. Use *ssh−add* to control the list of keys remembered by *ssh−agent*.
- When you run ssh, it contacts the ssh agent (over a named pipe) for the private key.

SSH Agent

- SSH Agent is *very* convenient. You can use ssh almost like a transparent rsh. Once keys are set up, you never have to type passphrases or login passwords.
- However, anyone who can get *root* on the machine running SSH Agent can get your private key.
- So *do not* use SSH Agent unless you control the machine and trust that no−one else has *root*.

SSH Agent Forwarding

- SSH Agent can even be forwarded over the SSH pipe.
- This means that SSH sessions on remote hosts can query the SSH Agent on your local host.
- This is even more dangerous than the normal use of SSH Agent. Don’t do it unless you trust all the machines along the way.

SCP

- SCP works just like RCP, but uses SSH for transport:

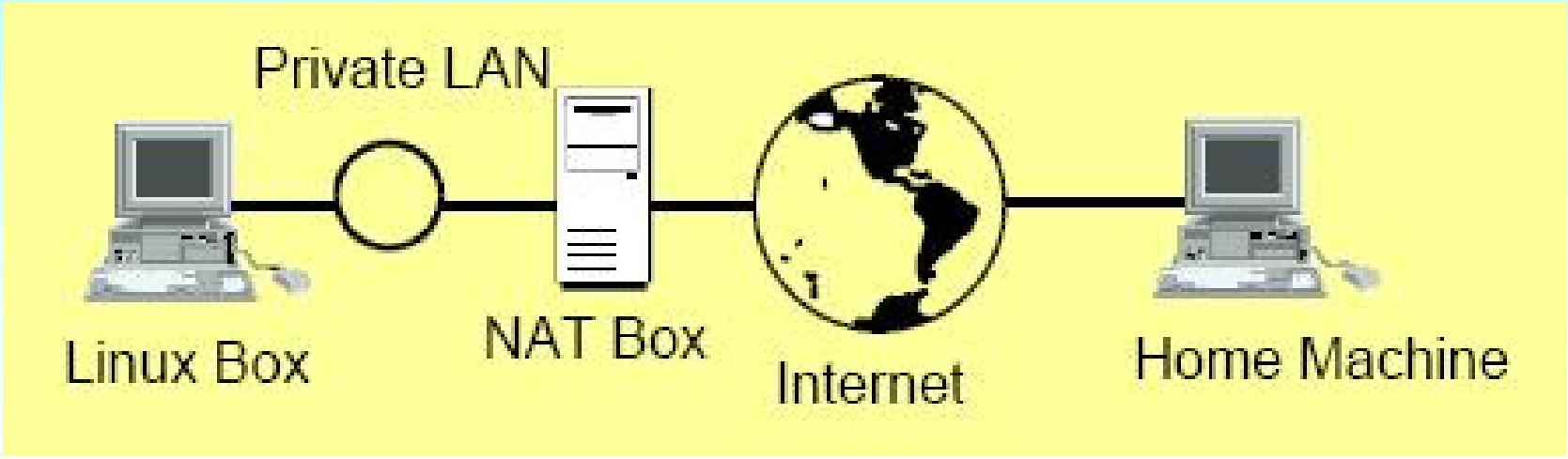
- scp localfile remotemach:/remote/file
- scp remotemach:/remote/file localfile
- scp file user@remote:/path

RSYNC over SSH

- RSYNC (http://rsync.samba.org) is a tool for efficient mirroring.
- It tries to copy as little as possible to make the remote side match the local side. It can often achieve "compression" ratios of 100−to−1.
- The latest rsync works reliably using the latest OpenSSH as its transport.

Firewall Busting

- Don’t try this at work.
- Many companies use a masquerading firewall (NAT-Network Address Translation) with unroutable IP addresses to limit access to internal networks.



Firewall Busting

- This kind of setup is *inconvenient*. There's no easy way to log on to your work Linux machine from home.
- Ahh, but... if you have a permanent or semi−permanent (or even non−permanent, if you are tricky) Internet connection at home, you can *bust through* the NAT box and log on to the Linux work machine.

Firewall Busting – Prep Work

- Install an SSH server on both your home and work machines. Have the servers start automatically at bootup.
- Write a script which runs on the work machine which periodically ssh's in to your *home* machine. It should simply run a "sleep 3600" command. Generate a key pair with no passphrase for the script to use.
- On your home machine, add the key to the authorized_keys list with a forced "sleep 3600" command.

Firewall Busting – The Magic

- Have the work machine include this argument to its ssh command: **–R 8822:localhost:22**
- Now the magic happens: Work machine calls up home machine. If authorized, executes sleep 3600 and port–forwards 8822 on home machine to port 22 on work machine.
- On home machine, ssh to localhost on port 8822. You'll be greeted with a login prompt from your work machine. You've busted through the NAT box.

Firewall Busting − Refinements

- NAT box limits you to certain ports? Run your home ssh server on port 80 (or 21 or whatever).
- Periodic connections are suspicious? Have work machine look for GPG−signed e−mail telling it to phone home. A fetchmail process can periodically check e−mail on your corporate server and kick in the ssh when it finds an appropriate signed e−mail.
- Moral: NAT doesn't solve everything. Covert channels are very hard to close.

SSH vs. IPSec

- SSH works at the application layer; IPSec works at the network layer. IPSec supported by big–name router companies.
- SSH simple to set up; IPSec more complicated.
- SSH can only forward TCP ports and doesn't work well with certain protocols (FTP); IPSec is a true VPN with transparent IP encryption.
- SSH protocol is simple; IPSec is complicated. *In general*, simplicity is preferred where security is at stake.

SSH vs. CIPE

- CIPE (Crypto IP Encapsulation) is a non−standard but very simple way of encrypting IP packets.
-
- Encapsulates IP in UDP.
- Much simpler than IPSec, but much less flexible. Intended for use between two routers.
- GPL'd Linux drivers; Windoze implementation under development.

Литература:

- http://www.wylug.org.uk/talks/2003/04/unix.pdf
- http://ce.sharif.edu/courses/ssc/unix/resources/root/Slides/unixhistory.pdf
- http://www.cs.uga.edu/~eileen/1730/Notes/intro-UNIX.ppt
- http://remus.rutgers.edu/cs416/F01
- http://www.cs.virginia.edu/~cs458/
- http://www.bobbooth.staff.shef.ac.uk/hpcs/materials/material.html
- http://www.comm.utoronto.ca/~jorg/teaching/ece461
- http://home.iitk.ac.in/~navi/sidbilinuxcourse/
- http://www.cs.washington.edu/homes/bershad/Mac/ssh/practicalmagic.pdf
- http://www.cs.cf.ac.uk/Dave/C/CE.html
- http://www.le.ac.uk/cc/tutorials/c/ccccintr.html
- http://www.shef.ac.uk/uni/academic/N-Q/phys/teaching/phy225/index.html